МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Івано-Франківський національний технічний університет нафти і газу

Кафедра комп'ютеризованого машинобудування

В. Б. Копей

МОВА ПРОГРАМУВАННЯ VBA ДЛЯ ІНЖЕНЕРІВ

НАВЧАЛЬНИЙ ПОСІБНИК

Івано-Франківськ
2019

УДК 004.43
К 65

Рецензент:
Панчук В. Г., доктор технічних наук, професор, завідувач кафедри комп'ютеризованого машинобудування Івано-Франківського національного технічного університету нафти і газу

К65 **Копей В. Б.** Мова програмування VBA для інженерів: Навчальний посібник / В. Б. Копей - Івано-Франківськ: ІФНТУНГ, 2019. - 125 с.

Навчальний посібник містить приклади програм мовою VBA з коментарями. Розглянуто основи програмування та використання COM-об'єктів для створення програм з графічним інтерфейсом користувача, роботи з офісними програмами Excel і Word, сервером сценаріїв Windows, САПР SOLIDWORKS, математичним пакетом MATLAB, доступу до даних тощо. Призначено для вивчення дисциплін "Основи програмування" та "Об'єктно-орієнтоване програмування", а також для виконання курсових і магістерських робіт під час підготовки бакалаврів і магістрів за спеціальністю 131 - Прикладна механіка.

УДК 004.43

ЗМІСТ

Вступ

VBA (Visual Basic for Applications) - об’єктно-орієнтована мова програмування загального призначення, яка розроблена Microsoft. VBA проста для вивчення, орієнтована на швидке розроблення програмного забезпечення (RAD) і часто використовується як мова сценаріїв (макросів) в різноманітних програмних продуктах (САПР, СКБД, офісних пакетах тощо), або як мова для системної інтеграції ("склеювання") різноманітних програмних компонентів (рис. 1).



Рисунок 1 - Особливості мови VBA

VBA основана на технології COM і дозволяє просто використовувати доступні COM-об’єкти та компоненти ActiveX. COM (Component Object Model) - платформа компонентно-орієнтованого програмування, яка використовується в ОС Windows. Підтримує повторене використання і можливість взаємодії об’єктів незалежно від мови програмування, на якій вони були розроблені.

Основними елементами COM є: об’єкт COM (екземпляр класу COM в сервері COM), сервер COM (програма, яка організовує доступ до створеного в ній об’єкта COM, реалізуючи інтерфейси), клієнт COM (програма, яка, використовуючи інтерфейс, отримує доступ до об’єкта COM), інтерфейс COM (визначає відкриті методи, які використовуються для доступу до об’єкта COM), клас COM (реалізація інтерфейсу COM в сервері COM). На COM основана технологія універсальних програмних компонентів ActiveX та технологія міжпроцесової взаємодії OLE Automation.

VBA вбудована як мова сценаріїв або має доступ до програмних інтерфейсів таких продуктів як Microsoft Office (Word, Excel, PowerPoint, Access тощо), CorelDRAW, Matlab, Maple, SOLIDWORKS, AutoCAD, FEMAP, CATIA V5, Autodesk Inventor, PTC Creo, SolidEdge. Завдяки цьому VBA часто використовується інженерами.

Більшість програм мовою VBA розробляються в зручному інтегрованому середовищі розробки (IDE) - редакторі Visual Basic, який має підказку коду, переглядач об’єктів, відлагоджувач, дизайнер графічного інтерфейсу користувача (GUI). Програма мовою VBA інтерпретується, а не компілюється.

VBA тісно пов’язана з мовою програмування Visual Basic і є її дещо спрощеною реалізацією. Також VBA має багато спільного з VBScript - мовою сценаріїв, що використовує технологію Active Scripting. Наступним кроком еволюції Visual Basic є Visual Basic .NET, що використовує сучасну платформу Microsoft .NET.

Недоліками VBA є те, що це пропрієтарне програмне забезпечення і не кросплатформове (в основному підтримується тільки Windows). Також VBA має дещо обмежені можливості у порівнянні з більш сучасними мовами сценаріїв (Python, Ruby, JavaScript), які розвиваються більш інтенсивно.

Основна мета цього посібника - швидке ознайомлення з основними можливостями VBA для створення прикладного інженерного програмного забезпечення. Розглянуто основи програмування, створення програм з графічним інтерфейсом користувача (GUI), використання COM-об'єктів для роботи з

офісними програмами (Excel, Word), сервером сценаріїв Windows, САПР (SOLIDWORKS, SOLIDWORKS Simulation), математичним пакетом MATLAB, доступу до даних (Microsoft DAO, Microsoft ADO, Msxml2.DOMDocument) та інше. Книга також може бути використана як короткий довідник з VBA. Посібник призначено для тих, хто уже володіє основами програмування якою-небудь алгоритмічною мовою. Паралельно з посібником автор рекомендує використовувати літературу [1-46] для глибшого освоєння матеріалу. Початківцям у першу чергу слід ознайомитись з книгами [19, 22, 25, 26, 30, 35, 36, 38, 39] для вивчення основ VBA або Visual Basic. Книги [5, 20, 29, 45] містять інформацію практично з усіх аспектів мови VBA і мають довідковий характер. Користуйтесь також вбудованою допомогою (клавіша F1) або online-довідником [37]. Для використання певної бібліотеки компонентів в вашій програмі клацніть на меню Tools/References... і виберіть цю бібліотеку (рис. 2). Для отримання довідкової інформації про бібліотеку або її компонент клацніть на меню View/Object Browser (або натисніть F2). В вікні Object Browser виберіть потрібний компонент і внизу з’явиться коротка довідка (рис. 3). Для детальної довідки виберіть Help в контекстному меню (рис. 3).

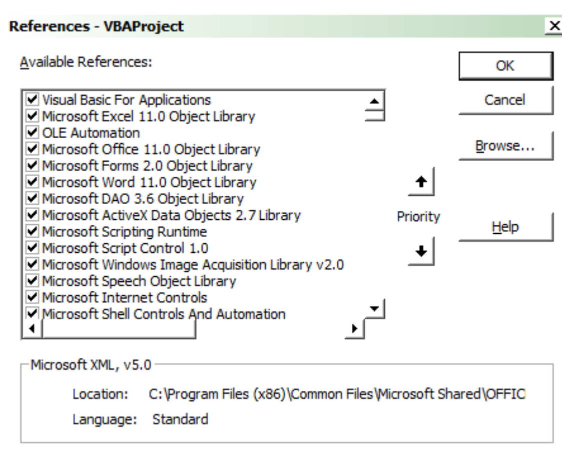


Рисунок 2 - Додання бібліотеки компонентів

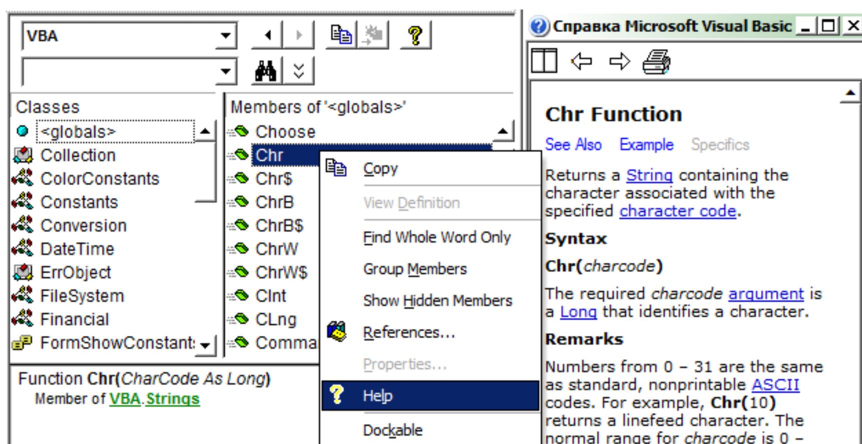


Рисунок 3 - Вікно Object Browser і довідка про функцію `Chr`

Приклади програм містять коментарі, що надруковані курсивом після символу `'` (апостроф). Ці коментарі не виконуються інтерпретатором. Код програм і результати їх виведення надруковані моноширинним шрифтом так:

```
код програми ' коментар
```

```
текст виведення програми
```

Вихідний код усіх прикладів доступний для вільного завантаження на GitHub (https://github.com/vkopey/VBA-for-engineers). В навчальному посібнику розглядаються версії VBA 6-7. Щоб розпочати вивчення VBA на комп’ютері достатньо мати установлений Excel 2003 або вище.

Автор буде вдячний читачам за зроблені зауваження і побажання, які можна залишити на сайті проекту в GitHub.

РОЗДІЛ 1. ОСНОВИ ПРОГРАМУВАННЯ МОВОЮ VBA

Найпростіша програма

Наступна програма виводить на екран `"Hello world!"`. Для створення програми відкрийте Excel і виберіть в меню Сервис/Макрос/Редактор Visual Basic (або натисніть Alt+F11). Щоб додати новий модуль виберіть в меню Insert/Module. Можна переіменувати модуль в "Hello" у вікні Properties (рис. 1.1). Щоб додати в модуль процедуру виберіть в меню Insert/Procedure... і назвіть її `main` (рис. 1.2).

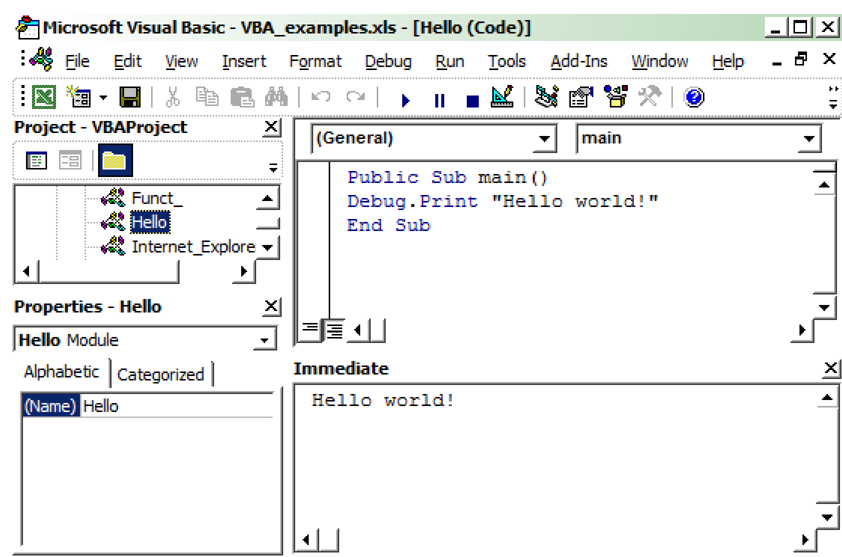


Рисунок 1.1 - Вигляд редактора Visual Basic

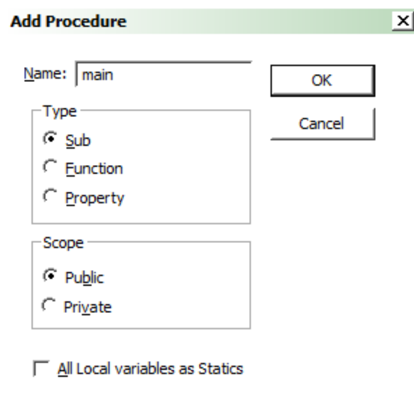


Рисунок 1.2 - Створення нової процедури `main`

З'явиться заготовка процедури без тіла. Введіть в процедуру код `Debug.Print "Hello world!"`:

```
Public Sub main()
Debug.Print "Hello world!"
End Sub
```

Виконайте програму через меню Run/Run Sub... (або натисніть F5). Якщо курсор знаходиться не в середині процедури `main`, то з'явиться вікно з запитом, де слід вибрати назву процедури, яку ви бажаєте виконати. Виберіть `main` і натисніть кнопку Run. Результат команди `Debug.Print` з'явиться в вікні Immediate (рис. 1.1)

Типи даних

Дане – поіменована область оперативної пам'яті, в якій зберігається значення певного типу. Даному характерні ім'я, адреса, тип і значення. Змінна – це дане, значення якого може змінюватися протягом виконання програми. Тип даних визначає допустимі значення даного, а також операції, які можуть виконуватись над

ним. Основними типами даних змінних в VBA є `Byte`, `Boolean`, `Integer`, `Long`, `Single`, `Double`, `Date`, `Object`, `String`, `Variant`. Користувач може визначати нові типи даних за допомогою оператора `Type`.

Як правило для оголошення змінних використовується оператор `Dim`. Якщо користувач не оголосив змінну, то її тип буде `Variant`. Наявність в модулі оператора `Option Explicit` вимагає явного оголошення усіх змінних модуля операторами `Dim`, `Private`, `Public`, `ReDim`, `static`. Змінна доступна там, де вона оголошена. Якщо змінна оголошена оператором `Public`, то вона загальнодоступна. Змінна, оголошена оператором `static` в процедурі, зберігає своє значення після виходу з процедури.

```
'Option Explicit
DefStr S 'змінні, назва яких починається з S, мають тип
string
'описати змінні з типом:
Dim i1 As Byte 'байт (коротке ціле від 0 до 255, розміром
1 байт)
Dim b As Boolean 'логічний (булевий) (значення: true (або
1), false (або 0))
Dim i2 As Integer 'цілий (ціле в межах +-32768, розміром 2
байти)
Dim i3 As Long 'довгий цілий (розміром 4 байти)
Dim x1 As Single 'дійсний звичайної точності (розміром 4
байти)
Dim x, y As Double 'дійсний подвійної точності (розміром 8
байт)
Dim d As Date 'календарна дата (розміром 8 байт)
Dim obj As Object 'об'єкт (розміром 4 байти)
Dim obj2 As New Worksheet 'об'єкт робочий лист Excel
Dim s As String 'рядок
Dim s2 As String * 10 'рядок розміром 10 символів
Dim x2 As Variant 'числові підтипи (розміром 16 байт)
Private Type student 'тип користувача, який описує поняття
студента -
```

```
    number As Integer 'його номер залікової книжки
    name As String 'і ім'я
End Type 'кінець опису типу
Dim obj3 As student 'описати змінну obj3 з типом student
Const s3 = "Hello!" 'константа
Public Const pi As Double = 3.14 'константа, видима в усіх
модулях

Public Sub main() 'підпрограма-процедура з іменем main,
'видима в усіх модулях проекту
'присвоїти змінним значення
i1 = 1 'ціле
b = True 'логічне
i2 = 12500 'ціле
i3 = 256132 'ціле
x1 = 5.124 'дійсне
x = 34.345 'дійсне
y = -25.684 'дійсне
d = Date 'присвоїти поточну дату, наприклад 21.09.2008
'присвоїти об'єкту obj вказівник на активну комірку Excel
Set obj = Excel.ActiveCell
obj.Value = 1 'присвоїти властивості Value значення 1
s = "hello world!" 'рядок
f$ = "Програмування на VBA" 'рядок
x2 = 54.76 'дійсне
x3 = 398 'ціле (змінна x3 не описана, тому її тип
Variant). Змінна буде не визначена, якщо забрати примітку
з Option Explicit на початку модуля.
obj3.number = 1 'полю number ціле
obj3.name = "Іванов" 'полю name рядок
'вивести значення даних у вікно Immediate
Debug.Print i1; b; i2; i3; x1; x; y; d; obj.Value; s; f$;
x2; x3; obj3.name
End Sub 'кінець підпрограми main
```

```
Зміна у $E$47
1 True 12500  256132  5,124  34,345 -25,684 25.08.2018  1
hello world!Програмування на VBA 54,76  398 Іванов
```

Відлагодження програм

Під час створення програм нерідко виникають помилки (bugs). Помилки поділяються на помилки компіляції (виникають під час трансляції програми або процедури), помилки часу виконання (виникають під час виконання програми) та логічні помилки (програма працює, але не так як потрібно). Помилки компіляції легко виявляються редактором коду або компілятором. Засобами виявлення помилок часу виконання і логічних помилок є: відлагодження (debug), засоби обробки виключних ситуацій мови програмування, зміна методики програмування, тестування програм. Засоби відлагодження доступні в меню Debug та на панелі Debug (рис. 1.3) [28, 29, 32]. За допомогою них можна призупинити/відновити виконання програми в потрібному місці (Toggle Breakpoint, Break, Run), виконувати програму в покроковому режимі (Step Into, Step Over, Step Out), переглядати значення змінних і виразів (Locals Window, Watch Window, Immediate Window), змінювати значення змінних (Watch Window), переглядати стек викликів (Call Stack).

Для ознайомлення з Debug виконайте наступні дії:

1. Створіть модуль з попереднім прикладом.
2. Поставте точку зупинки в довільному місці процедури `main` (Toggle Breakpoint).
3. Запустіть програму (Run).
4. Перегляньте вміст вікон на рис. 1.3.
5. Змініть значення довільної змінної у вікні Watches.
6. Виконайте кілька кроків програми (Step Into або F8).
7. Виконайте програму до кінця (Run).
8. Перегляньте вміст вікна Immediate.

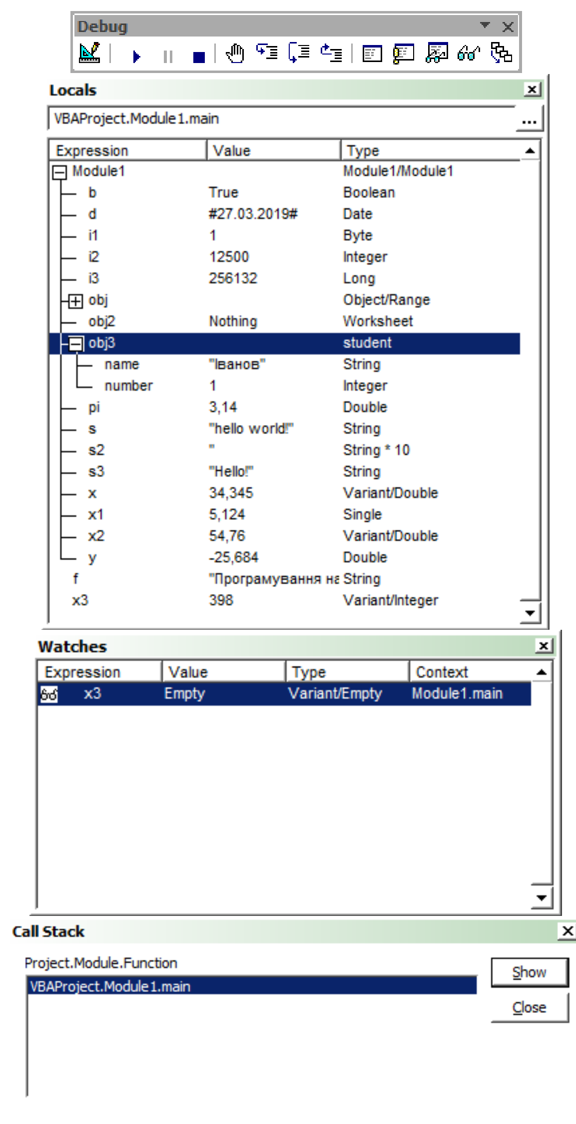


Рисунок 1.3 - Панель Debug і вікна Locals, Watches, Call Stack

Арифметичні оператори

В прикладі показані найбільш уживані арифметичні оператори: додавання, віднімання, множення, ділення тощо. Ви можете переглядати значення змінної "c" в режимі відлагодження у вікні Watches.

```
Dim A, b, c As Double

Public Sub main()
A = 3: b = 2
c = A + b 'додавання c=5
c = A - b 'віднімання c=1
c = -A 'зміна знаку c=-3
c = A * b 'множення c=6
c = A / b 'ділення c=1,5
c = A \ b 'цілочисельне ділення c=1
c = A Mod b 'остача від ділення c=1
c = A ^ b 'піднесення до степеня c=9
End Sub
```

Оператори порівняння і логічні оператори

В прикладі показано використання операторів порівняння (<, >, >= тощо) і логічних операторів (And, Or, Not тощо). Ці оператори використовуються в логічних виразах, результатом яких є значення типу Boolean (True або False). Ви можете переглядати значення змінної c в режимі відлагодження у вікні Watches.

```
Dim x, y As Double
Dim A, b, c As Boolean

Public Sub main()
x = 2: y = 3
A = True: b = False
```

```
c = x > y 'більше: c=False
c = x < y 'менше: c=True
c = x >= y 'більше дорівнює: c=False
c = x <= y 'менше дорівнює: c=True
c = x <> y 'не дорівнює: c=True
c = x = y 'дорівнює: c=False
c = A And b 'логічне "І": c=False
c = A Or b 'логічне "АБО": c=True
c = Not A 'логічне "НЕ": c=False
c = A Xor b 'виключна диз’юнкція: c=True
End Sub
```

Пріоритет операторів

Пріоритетом операторів називають порядок їх виконання в складних виразах. Спочатку виконуються арифметичні оператори, потім - оператори порівняння, а потім - логічні оператори. Якщо оператори мають однаковий пріоритет, то вони виконуються зліва направо. Пріоритет операторів:

- виклик функції і дужки;
- ^;
- - (зміна знаку);
- *, /;
- \;
- Mod;
- +, -;
- оператори порівняння (мають однаковий пріоритет);
- логічні оператори (в порядку Not, And, Or, Xor, Eqv, Imp).

```
Dim x, y As Double
Dim b As Boolean

Public Sub main()
x = 1: y = 2
'обчислення цього виразу виконується за правилом
```

```
пріоритету операторів
b = (x ^ (2 + x) + 1) / (x * Cos(x + 1) - x) + x = 1 Or
y > 0
Debug.Print b
'And має вищий пріоритет, Or - нижчий
Debug.Print True Or False And False ' True
Debug.Print (True Or False) And False ' False
End Sub
```

```
True
True
False
```

Оператор безумовного переходу GoTo

Виконання оператора безумовного переходу `GoTo <мітка>` призводить до переходу на рядок процедури, який позначений міткою `<мітка>`. Не рекомендується часто використовувати цей оператор у складних алгоритмах, оскільки це ускладнює їх розуміння і відлагодження.

```
Public Sub main()
x = 1
y = 2
If x = 1 Then GoTo 10 'якщо x=1, перейти на рядок 10
y = 3
10 Debug.Print y 'рядок з міткою 10
End Sub
```

```
2
```

Оператор умови If-Then-Else

Оператор умови `If <умова> Then <команди>` виконує `<команди>` тільки тоді, коли значення логічного виразу `<умова>` рівне `True` (істина). Інструкція `If` може застосовуватись з `ElseIf` та/або `Else`. Якщо значення логічного виразу рівне `False` (не істина), то виконується інструкція `ElseIf`, або, якщо її немає,

виконується `Else`. Послідовних інструкцій `ElseIf` може бути довільна кількість. Кінець блоку `If` позначається `End If`.

```
Dim A, x, y As Double

Public Sub main()
x = 2: A = 1

'______________1-й варіант конструкції______________
If x > 0 Then y = x + A 'якщо x>0, то y=x+a

'______________2-й варіант конструкції______________
If x > 0 Then y = x + A Else y = x - A 'якщо x>0, то
y=x+a, інакше y=x-a

'______________3-й варіант конструкції______________
If x > 0 Then 'якщо x>0, то
A = 2
y = x + A
Else 'інакше
A = 1
y = x - A
End If 'кінець умови

'______________4-й варіант конструкції______________
If x > 0 Then 'якщо x>0, то
A = 2
y = x + A
ElseIf x < 0 And x >= -5 Then 'інакше, якщо x<0 і x>=-5,
то
A = 1
y = x - A
Else 'інакше
y = 0
End If 'кінець умови
End Sub
```

Оператор вибору Select Case

Оператор вибору `Select Case <змінна>` дозволяє виконувати певні команди в залежності від значення змінної `<змінна>`. Якщо змінна рівна певному значенню, що перевіряється командою `Case <умови>`, то виконуються команди нижче. Якщо змінна не рівна жодному значенню, то виконуються команди після слів `Case Else`. Цей оператор дозволяє уникати застосування складних інструкцій `If-ElseIf`.

```
Dim x, y As Double

Public Sub main()
x = 2.7
Select Case x 'вибір стосується змінної 'x'
Case 1.5 'якщо x=1.5, то
y = 7.4
Case 2 To 2.7, 3, Is > 4 'якщо 2<=x<=2.7 або x=3 або x>4,
то
y = 3.2
Case Else 'у інших випадках
y = 0
End Select 'кінець вибору
Debug.Print y
End Sub
```

Функції вибору

Функції вибору (`IIf`, `Choose`, `Switch`) призначені для вибору значення зі списку в залежності від заданих умов. В деяких випадках вони дозволяють суттєво зменшити об’єм коду програми шляхом уникнення застосування операторів `If-Then-Else` та `Select Case`.

```
Dim x, y As Double
Dim i As Integer
```

```
Public Sub main()
x = 1.5: i = 2
y = IIf(x = 1.5, 6, 0) 'якщо умова істинна, то повертає 6, інакше 0
y = Choose(i, 4.6, 3.7, 8, 10.2) 'якщо i=1, то y=4.6, якщо i=2, то y=3.7, і т.д.
y = Switch(x = 0.5, 0.2, x = 1.5, 4.1) 'якщо x=0.5, то повертає 0.2, якщо x=1.5, то повертає 4.1
End Sub
```

Оператори циклу For-Next

Оператори циклу призначені для циклічного виконання заданого блоку команд. Оператор циклу `For-To-Next` циклічно виконує блок команд, поки змінна циклу не досягне заданого значення. На кожній ітерації змінна циклу збільшує своє значення на 1, але дозволяється вказувати довільний крок після слова `Step`. Оператор циклу `For-Each-Next` циклічно виконує блок команд, для кожного елемента в масиві або колекції.

```
Dim i, s As Integer
Dim x, y As Double
Dim obj As Object

Public Sub main()
'Приклад 1: знайти суму цілих чисел від 1 до 100
s = 0 'присвоїти сумі 0
For i = 1 To 100 'i змінюється від 1 до 100
     s = s + i 'додати до суми 'i'
Next i 'наступне 'i'
Debug.Print s 'вивести

'Приклад 2: вивести таблицю значень y=Sin(x), якщо 'x' змінюється від 0 до 1 з кроком 0.1
For x = 0 To 1 Step 0.1 'x змінюється від 0 до 1 з кроком
```

```
0.1
    y = Sin(x) 'обчислення функції
    Debug.Print "x="; x; "y="; y 'вивести
Next x 'наступне 'x'

'Приклад 3: знайти індекс елемента масиву зі значенням 7
A = Array(2, 5, 7, 1) 'масив
For i = 0 To 3 'i змінюється від 0 до 3
    'якщо елемент рівний 7, то вивести індекс і вийти з
циклу
    If A(i) = 7 Then Debug.Print i: Exit For
Next i 'наступне 'i'

'Приклад 4: вивести елементи масиву A
For Each x In A 'для кожного x у A
    Debug.Print x; 'вивести
Next x 'наступний x

'Приклад 5: вивести в комірки A1:A5 1
For Each obj In Range("A1:A5") 'для кожного об'єкта obj у
діапазоні комірок
    obj.Value = 1 'присвоїти значенню комірки 1
Next obj 'наступний об'єкт
End Sub
```

```
5050
x= 0 y= 0
x= 0,1 y= 9,98334166468282E-02
x= 0,2 y= 0,198669330795061
x= 0,3 y= 0,29552020666134
x= 0,4 y= 0,389418342308651
x= 0,5 y= 0,479425538604203
x= 0,6 y= 0,564642473395035
x= 0,7 y= 0,644217687237691
x= 0,8 y= 0,717356090899523
x= 0,9 y= 0,783326909627483
x= 1 y= 0,841470984807896
 2
```

```
 2   5   7   1 Зміна у $A$1
Зміна у $A$2
Зміна у $A$3
Зміна у $A$4
Зміна у $A$5
```

Оператор циклу While-Wend

Оператор циклу `While-Wend` (цикл з передумовою) циклічно виконує блок команд, поки значення заданого логічного виразу істинне. Спочатку обчислюється значення логічного вразу, а потім виконується блок команд, який завершується словом `Wend`.

```
Dim i, s As Integer

Public Sub main()
'знайти суму цілих чисел від 1 до 100
s = 0 'сума = 0
i = 1 'перше число
While i <= 100 'поки i менше рівне 100
s = s + i 'додати до суми 'i'
i = i + 1 'наступне 'i'
Wend 'повторити
Debug.Print s 'вивести суму
End Sub
```

```
5050
```

Оператори циклу Do-Loop

Оператор циклу `Do-Loop-While` (цикл з післяумовою) циклічно виконує блок команд, поки значення заданого логічного виразу істинне. Спочатку виконується блок команд, а потім обчислюється значення логічного вразу.

Оператор циклу `Do-While-Loop` (цикл з передумовою) циклічно виконує блок команд, поки значення заданого логічного

виразу істинне. Спочатку обчислюється значення логічного вразу, а потім виконується блок команд, який завершується словом Loop.

```
Dim i, s As Integer

Public Sub main()
'Приклад 1: знайти суму цілих чисел від 1 до 100
s = 0 'сума = 0
i = 1 'перше число
Do 'виконувати цикл
s = s + i 'додати до суми 'i'
i = i + 1 'наступне 'i'
Loop While i <= 100 'повторювати, поки 'i' менше рівне 100
Debug.Print s 'вивести суму

'Приклад 2: знайти суму цілих чисел від 1 до 100
s = 0 'сума = 0
i = 1 'перше число
Do While i<=100 'виконувати цикл, поки 'i' менше рівне 100
s = s + i 'додати до суми 'i'
i = i + 1 'наступне 'i'
Loop  'повторити
Debug.Print s 'вивести суму
End Sub
```

```
5050
5050
```

Перехоплення помилок часу виконання

Оператор On Error дозволяє перехоплювати помилки часу виконання (ділення на нуль, переповнення та інші). Команда Debug.Assert <логічний вираз> призначена для зупинки виконання програми на рядку з цією командою, якщо значення логічного виразу рівне False. Продовжити виконання програми можна в звичайному (кнопка Run або клавіша F5) або покроковому режимі. В прикладі перехоплюються різні помилки і виводиться інформація про них за допомогою об’єкта Err. Спробуйте

закоментувати рядок `On Error GoTo HandleError`, щоб побачити різницю.

```
Dim x As Double

Public Sub main()
x = 0
Debug.Assert x <> 0 'якщо 'x' дорівнює 0, призупинити виконання
On Error GoTo HandleError 'при помилці перейти на мітку HandleError
Debug.Print 1 / x 'помилка 11 (ділення на нуль)
x = 1E+300 * 1E+300 'помилка 6 (переповнення)
x = CDbl("0.12") 'помилка 13 (невідповідність типу)
Err.Raise 65535 'створити помилку виконання 65535
Exit Sub 'вийти з підпрограми
HandleError: 'мітка
Select Case Err.number 'якщо номер помилки
    Case 11 'рівний 11 (ділення на нуль)
        x = 1 'змінити знаменник
        Debug.Print "Ділення на нуль!" 'вивести повідомлення
        Resume 'повторити інструкцію з помилкою
    Case Else 'інший номер
        Debug.Print Err.number 'вивести номер помилки
        Debug.Print Err.Description 'вивести опис помилки
End Select
Resume Next 'перейти на наступну інструкцію за помилкою
End Sub
```

```
Ділення на нуль!
 1
 6
Overflow
 13
Type mismatch
 65535
Application-defined or object-defined error
```

Підпрограма-процедура Sub

Процедура - це іменована частина коду програми (підпрограма), до якого можна звертатись з інших частин програми. Описується процедура за допомогою оператора `Sub`, після якого вказується назва процедури і в дужках описуються її параметри (аргументи). Опис процедури завершується словами `End Sub`. Викликати процедуру можна так: `<назва> <аргумент1>, <аргумент2>`. За замовчуванням аргументи передаються в процедуру за посиланням (`ByRef`). Для прикладу значення змінної `x`, яка передана в процедуру за посиланням, може бути змінене в процедурі. Але можна передавати аргументи за значенням (`ByVal`). В цьому випадку в процедурі буде створена окрема копія змінної `x`, і зміна її значення не вплине на `x`.

Процедуру, описану словом `Public`, можна викликати з будь-якого модуля. Змінні, які описані в процедурі оператором `Dim` є локальними, тобто доступні тільки в цій процедурі.

```
Dim x, y As Integer 'глобальні змінні
Dim B1(0 To 2), B2(2), B3(2) As Integer 'глобальні масиви

'Головна підпрограма-процедура main
Public Sub main()
'Приклад 1
Sum 2, 3, y 'виклик процедури Sum з параметрами 2, 3, y
'або Call Sum(2, 3, y)
'або Sum A:=2, B:=3, C:=y
Debug.Print y 'виведення 'y'

'Приклад 2
x = 1: y = 1
Sum2 x, y 'виклик процедури Sum2. Результат: x=3, y=1
Debug.Print x; y 'виведення

'Приклад 3
```

```
B1(0) = 1: B1(1) = 5: B1(2) = 3 'заповнити масив B1
B2(0) = 9: B2(1) = 5: B2(2) = 7 'заповнити масив B2
Sum3 B1, B2, B3 'виклик процедури Sum3
Debug.Print B3(0), B3(1), B3(2) 'виведення

'Приклад 4
Sum4 2, 3 'виклик процедури Sum4

'Приклад 5
Sum5 1, x, y 'виклик процедури Sum5
Debug.Print x, y 'виведення
End Sub 'кінець процедури

'Процедура Sum
Public Sub Sum(A, b, c As Integer) 'a, b, c - параметри
c = A + b 'тіло процедури
End Sub 'кінець процедури

'Процедура Sum2
'Параметр A передається за посиланням (за замовчуванням),
B - за значенням
Public Sub Sum2(ByRef A As Variant, ByVal b As Integer)
Dim n As Integer 'локальна змінна
n = 2
A = A + n 'A - синонім 'x'
b = b + n 'B - окрема копія 'y'
End Sub

'Процедура Sum3
Public Sub Sum3(A1(), A2(), A3() As Integer) 'параметри -
масиви
For i = 0 To 2
    A3(i) = A1(i) + A2(i) 'додати масиви A1 і A2
Next i
End Sub

'Процедура Sum4
'Параметр C не обов'язковий, за замовчуванням рівний 1
```

```
Public Sub Sum4(A, b As Integer, Optional c As Integer =
1)
'якщо не вказано С і A=0, вийти з процедури
If IsMissing(c) And A = 0 Then Exit Sub
Debug.Print A + b + c 'вивести суму
End Sub

'Процедура Sum5
Public Sub Sum5(A As Integer, ParamArray z()) 'необмежена
кількість параметрів
For i = LBound(z) To UBound(z)
    z(i) = z(i) + A 'додати до кожного параметра A
Next i
End Sub
```

```
 5
 3  1
 10            10            10
 6
 4             2
```

Підпрограма-функція Function

Підпрограма-функція (або просто функція) подібна на процедуру Sub, але описується за допомогою слова Function і може повертати значення. Для прикладу щоб функція f(x) повертала значення x^2 в її тілі повинна бути команда f=x*x. Якщо таку функцію викликати командою y=f(2), то змінній y буде присвоєно 4.

```
Dim y As Integer

'Головна підпрограма-процедура main
Public Sub main()
y = Sum(2, 3) 'присвоїти 'y' значення функції Sum з
параметрами 2, 3
Debug.Print y 'вивести 'y'
Debug.Print Sum2() 'результат: 1
```

```
Debug.Print Sum2() 'результат: 2
Debug.Print Sum2() 'результат: 3
Debug.Print Fact(3) 'вивести факторіал 3
End Sub 'кінець процедури

'Функція Sum, яка повертає значення цілого типу
Public Function Sum(A, b As Integer) As Integer 'a, b -
параметри
Sum = A + b 'повернути суму a + b
End Function 'кінець функції

'Функція з статичною локальною змінною
Public Function Sum2() As Integer
Static n As Integer 'статична змінна зберігає своє
значення
n = n + 1 'змінити значення статичної змінної
Sum2 = n 'повернути N
End Function

'Рекурсивна функція (викликає сама себе) для обчислення
факторіала
Public Function Fact(n As Integer)
'якщо N < 1, то Fact = 1, інакше викликати Fact(N - 1) і
помножити на N
If n < 1 Then Fact = 1 Else Fact = Fact(n - 1) * n
End Function
```

```
5
1
2
3
6
```

Масиви

Масив - це іменований впорядкований набір даних одного типу. Масив складається з елементів, кожен з яких доступний за унікальним індексом цілого типу. Масиви можуть бути одновимірні

і багатовимірні. Оголошуються масиви так само як звичайні змінні, але після назви масиву потрібно вказати його максимальний індекс. Наприклад `Dim A(2)`. Масив `A` буде містити 3 елемента. Якщо в дужках нічого не вказано, наприклад `Dim A()`, то такий масив називається динамічним. Індексація масиву починається з 0, якщо в програмі не вказано `Option Base 1`. Звернутись до елементів масиву `A` можна так: `A(0)`, `A(1)`, `A(2)`.

```
Option Base 0 'індексація масивів з нуля
Dim A(3) As Byte 'масив A коротких цілих розміром 4
Dim b(1 To 2) As Double 'масив B дійсних розміром 2
Dim c(1 To 2, 1 To 2) As Integer 'масив C цілих розміром
2x2
Dim d() As Boolean 'динамічний масив D логічних
Dim E As Variant 'змінна або масив E типу Variant

'підпрограма main
Public Sub main()
A(0) = 7 'присвоїти елементу масиву A з індексом 0 число 7
A(1) = 9 '-//- 1 число 9
A(2) = 2 '-//- 2 число 2
Debug.Print "A()="; A(0); A(1); A(2) 'вивести A
b(1) = 3.56 'заповнити масив B
b(2) = 5.78
Debug.Print "B()="; b(1); b(2) 'вивести B
c(1, 1) = 5 'присвоїти елементу масиву C з індексами 1,1
число 5
c(1, 2) = 10 '(рядок 1, стовпчик 2) число 10
c(2, 1) = 20
c(2, 2) = 15
Debug.Print "C()="; c(1, 1); c(1, 2) 'вивести C
Debug.Print "    "; c(2, 1); c(2, 2)
ReDim d(1 To 2) 'змінити розмір динамічного масиву D
d(1) = 1 'заповнити масив D
d(2) = 0
Erase d 'вивільнити пам'ять для динамічного масиву D
ReDim d(1 To 2, 1 To 2) 'змінити розмір динамічного масиву
```

```
D
d(1, 1) = 1 'заповнити масив D
d(1, 2) = 1
d(2, 1) = 0
d(2, 2) = 0
Debug.Print "D()="
'вкладені цикли для виведення масиву D
For i = LBound(d, 1) To UBound(d, 1) 'від першого рядка до
останнього
    For j = LBound(d, 2) To UBound(d, 2) 'від першого
стовпчика до останнього
        Debug.Print d(i, j); " "; 'вивести значення
елемента D(i, j)
    Next j 'наступний стовпчик
    Debug.Print
Next i 'наступний рядок
E = Array(2, 1, 5, 6, 0) 'заповнити варіантний масив E
'Debug.Print 'перейти на наступний рядок
Debug.Assert IsArray(E)  'якщо E не є масивом, призупинити
виконання програми
Debug.Print "E()="; E(0); E(1); E(2); E(3); E(4) 'вивести
E
End Sub 'кінець підпрограми main
```

```
A()= 7  9  2
B()= 3,56  5,78
C()= 5  10
     20  15
D()=
True True
False False
E()= 2  1  5  6  0
```

Колекції Collection

Колекція (Collection) - це об’єкт, який містить впорядкований набір елементів, що можуть мати довільний тип. Звернутись до елемента колекції можна за його унікальним індексом (ключем).

Колекція містить методи, які дозволяють додавати нові елементи, вставляти їх в задане положення, видаляти і підраховувати їх кількість. Зазвичай використовувати колекції зручніше ніж масиви. В наступному прикладі використовуйте `Set c = Nothing` для видалення колекції, або описуйте колекцію всередині процедури `main`.

```
Dim c As New Collection 'колекція
Dim x As Variant

Public Sub main()
c.Add "січень" 'додати елемент з ключем 1
c.Add "лютий" 'додати елемент з ключем 2
c.Add "грудень", before:=1  'вставити елемент перед 1
c.Add "березень", "5" 'додати елемент з ключем 5
c.Remove 2 'видалити елемент з ключем 2 ("січень")
Debug.Print c.Count 'кількість елементів
Debug.Print c(1), c(2), c("5") 'значення елементів за
ключами
'Debug.Print c.Item("5")'або так
For Each x In c 'для всіх елементів 'x' в колекції 'c'
    Debug.Print x, 'вивести елемент
Next x
Set c = Nothing
End Sub
```

```
3
грудень       лютий         березень
грудень       лютий         березень
```

Словник Scripting.Dictionary

Словник (`Scripting.Dictionary`) - це об’єкт, який містить дані у вигляді множини пар ключ-значення. Значення можуть мати довільний тип. Звернутись до значення словника можна за його унікальним ключем. Словник містить властивості і методи, які дозволяють додавати і видаляти дані, змінювати ключі, перевіряти

наявність ключів та інші. Для запуску прикладу під'єднайте бібліотеку Microsoft Scripting Runtime.

```
Dim d As New Scripting.Dictionary 'словник
Dim k As Variant

Public Sub main()
'додати в словник елемент з ключем 1 і значенням "Січень"
d.Add 1, "Січень"
d.Add 2, "Лютий"
d.Add 3, "Березень"
For Each k In d 'для кожного ключа 'k' в словнику 'd'
    Debug.Print k, d(k) 'вивести ключ і значення
    'Debug.Print k, d.Item(k) 'або так
Next k
d.Remove 2 'видалити елемент з ключем 2
Debug.Print d.Exists(2) 'чи існує елемент із ключем 2
d.Key(3) = 2 'змінити ключ
Debug.Print d.Count 'вивести кількість елементів
Debug.Print d.keys(0), d.items(0) 'перші елементи масивів ключів і значень
Set d = Nothing
End Sub
```

```
 1            Січень
 2            Лютий
 3            Березень
False
 2
 1            Січень
```

Функції перетворення типів

В прикладі показано використання функцій для перетворення даних з одного типу в інший. Зокрема можна перетворити рядок в число `Val("3.51")` або число в рядок `Str(3.51)`.

```
Dim x As Double
```

```
Dim s As String
Dim b As Boolean
Dim d As Date
Dim i As Integer

Public Sub main()
x = Val("3.51") 'конвертує рядок в число, x=3.51
s = Str(x) 'конвертує число в рядок, s="3.51"
s = Format(Date, "ddd, d mm yyyy") 'форматує дані, s="Вт,
30 09 2008"
s = Format(Date, "dddd, d mm yyyy") 's="вторник, 30 09
2008"
s = Format(15790.335, "##,##0.00 грн") 's="15 790,34 грн"
(тип Variant)
s = Format$(15790.335, "##,##0.00 грн") 's="15 790,34 грн"
(тип String)
b = CBool("false") 'конвертує в булеве, b=false
d = CDate("30.09.2008") 'конвертує в дату, d=#30.09.2008#
x = CDbl("3,51") 'конвертує в дійсне, x=3.51
i = CInt("15") 'конвертує  в ціле, i=15
s = CStr(3.51) 'конвертує в рядок, s="3.51"
y = CVar(15 & "00") 'конвертує у Variant, y=1500
End Sub
```

Математичні функції

Приклад показує роботу з різними математичними функціями (Abs, Cos, Sqr тощо). Зверніть увагу, що тригонометричні функції потребують значення кута в радіанах.

```
Dim x, y As Double
Dim i As Integer

Public Sub main()
x = 0.5
y = Abs(x) 'модуль
y = Atn(x) 'арктангенс
```

```
y = Cos(x) 'косинус кута в радіанах
y = Exp(x) 'експонента (приблизно 2.71^x)
y = Log(x) 'натуральний логарифм
y = Rnd(x) 'випадкове число від 0 до 1
y = Sgn(x) 'повертає 0, 1, -1 в залежності від знаку 'x'
y = Sin(x) 'синус кута в радіанах
y = Sqr(x) 'корінь квадратний
y = Tan(x) 'тангенс кута в радіанах
i = Fix(x) 'повертає найближче найменше ціле для додатного
числа і найближче найбільше ціле для від'ємного
i = Int(x) 'повертає найближче найменше ціле
'похідні математичні функції
y = 1 / Tan(x) 'котангенс
y = Atn(x / Sqr(-x * x + 1)) 'арксинус
y = Atn(-x / Sqr(-x * x + 1)) + 2 * Atn(1) 'арккосинус
y = Atn(x) + 2 * Atn(1) 'арккотангенс
y = Log(x) / Log(5) 'логарифм з основою 5
y = Int((9 - 5 + 1) * Rnd + 5) 'випадкове число від 5 до 9
End Sub
```

Функції обробки рядків

В прикладі показані різні функції для обробки рядків (тип String). Ці функції дозволяють об'єднувати рядки, обчислювати їх довжину, шукати текст тощо.

```
Dim s1, s2 As String

Public Sub main()
c = Asc("Y") 'повертає ASCII код символу
s1 = Chr(89) 'повертає символ за ASCII кодом
'Chr(13) - символ переходу на новий рядок
Debug.Print "hello" & Chr(13) & "world"
s1 = "Hello World!!!"
s2 = Mid(s1, 1, 5) 'повертає підрядок, починаючи з 1-го
символу, довжиною 5 символів, s2="Hello"
i = Len(s1) 'довжина рядка, i=14
```

```
s1 = " " & s2 & "  " 'додати до рядка символи пробілу
s2 = Trim(s) 'видаляє символи пробілу спочатку і вкінці
рядка
i = InStr(s1, s2) 'повертає позицію першого входження s2 у
s1, i=2
i = StrComp(s1, s2) 'порівнює рядки, i=1 (s1 > s2)
End Sub
```

```
hello
world
```

Функції обробки дати і часу

Функції обробки дати і часу дозволяють визначати дату і час, визначати компоненти дати, розраховувати різницю дат, додавати до дати інтервали. Ці функції працюють зі змінними типу `Date`.

```
Dim d, T As Date
Dim i As Integer
Dim l As Long
Dim f As Single

Public Sub main()
d = Date 'поточна дата, d=#9/30/2008#
T = Time 'поточний час, d=#20:31:07#
d = Now 'поточні дата і час, d=#9/30/2008 20:31:07#
d = #9/30/2008# 'присвоїти дату
T = #8:31:07 PM# 'присвоїти час
d = #9/30/2008 8:31:07 PM# 'присвоїти дату і час
d = DateSerial(2008, 9, 30) 'дата, задана цілими числами
T = TimeSerial(20, 31, 7) 'час, заданий цілими числами
i = Day(d) 'день
i = Month(d) 'місяць
i = Year(d) 'рік
i = Hour(T) 'година
i = Minute(T) 'хвилина
i = Second(T) 'секунда
i = Weekday(T) 'день тижня
f = Timer 'число секунд після опівночі
```

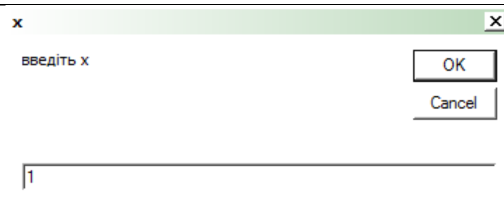
```
l = DateDiff("h", #8/12/1978#, Now) 'кількість інтервалів
між датами (годин)
i = DatePart("m", Date) 'компонент дати (місяць)
d = DateAdd("m", 10, Date) 'додає до дати інтервал (10
місяців)
T = TimeValue("3:31:30 PM") 'перетворює рядок в формат
часу
End Sub
```

Функції для створення діалогових вікон

В прикладі показані функції для створення діалогових вікон (рис. 1.4). Функція InputBox створює діалогове вікно для введення даних. Функція MsgBox створює діалогове вікно з повідомленням і кнопками, а також повертає код натиснутої кнопки.

```
Dim x As Double
Dim i As Byte

Public Sub main()
x = InputBox("введіть x", "x", 0) 'виводить діалогове
вікно з полем введення і повертає введене значення
i = MsgBox("x=" & x, vbYesNoCancel, "Аргумент") 'виводить
діалогове вікно з повідомленням і кнопками Yes/No/Cancel
MsgBox (i) 'код натиснутої кнопки
End Sub
```



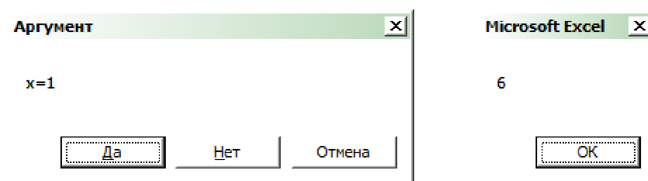


Рисунок 1.4 - Діалогові вікна

Використання функцій з бібліотек DLL

В програмі мовою VBA можуть бути використані довільні функції з бібліотек DLL, в тому числі функції Win32 API [20, 21, 28, 29, 42]. Наприклад можна відтворити звуковий файл за допомогою функції `PlaySound` з системної бібліотеки winmm.dll.

```
'функція MessageBeep з системної бібліотеки user32.dll
Declare Sub MessageBeep Lib "user32.dll" (ByVal T As Long)
'функція PlaySound з системної бібліотеки winmm.dll
Declare Function PlaySound Lib "winmm.dll" Alias
"PlaySoundA" _
(ByVal n As String, ByVal m As Long, ByVal f As Long) As
Long

Public Sub main()
MessageBeep 32 'виклик функції з параметром
'виклик функції з параметром
Call PlaySound("c:\WINDOWS\Media\tada.wav", 0&, SND_ASYNC
Or SND_FILENAME)
End Sub
```

Файли послідовного доступу

Файл - це інформаційний об’єкт, який містить послідовність байтів і розміщений у файловій системі на носію інформації. Усі файли є бінарними, але якщо для файлу застосовується кодування символів (ASCII, UTF-8, CP1251 або інше), то його називають

текстовим. Файли послідовного доступу, наприклад текстові, зберігають дані в неструктурованому для звернення вигляді. Для пошуку в таких файлах потрібно послідовно читати усі дані. Для роботи з файлом його відкривають оператором `Open` і вказують: шлях до файлу, режим (`Append`, `Binary`, `Input`, `Output`, `Random`) і номер файлу. Для запису даних у файл послідовного доступу використовують оператори `Print` і `Write`, а для читання - `Input` та `Line Input`. Після завершення роботи з файлом його потрібно закрити оператором `Close`. Приклад створює текстовий файл послідовного доступу, записує в нього дані, читає дані та додає нові дані.

```
Dim x, y As Double
Dim s As String

'підпрограма main
Public Sub main()
x = 5.6
'відкрити файл c:\file1.dat під номером 1 для виведення
Open "c:\file1.dat" For Output As #1
Print #1, "значення x="; x 'записати форматовані дані у
файл 1
Write #1, x ^ 2 'записати неформатовані дані у файл 1
Close #1 'закрити файл 1

'відкрити файл c:\file1.dat для введення під номером 1
Open "c:\file1.dat" For Input As #1 '
Line Input #1, s 'прочитати рядок з файлу
Input #1, y 'прочитати дане з файлу
Debug.Print s; y 'вивести s,y у вікно Immediate
Close #1 'закрити файл 1

'відкрити файл c:\file1.dat під номером 1 для додання
Open "c:\file1.dat" For Append As #1
Write #1, x ^ 3 'записати неформатовані дані у файл 1
Close #1 'закрити файл 1
```

```
End Sub
```

Бінарні файли

Приклад відкриває файл як бінарний і читає його побайтово за допомогою функції `Input`. Кінець файлу визначається за допомогою функції `EOF` (End Of File).

```
Public Sub main()
'відкрити файл c:\file1.dat під номером 1 як бінарний
Open "c:\file1.dat" For Binary As #1
Do While Not EOF(1) 'поки не кінець файлу 1
    x = Input(1, #1) 'присвоїти 'x' наступний байт з файлу
1
    Debug.Print x; 'вивести 'x'
Loop 'повторити
Close #1 'закрити файл 1
End Sub
```

Файли довільного доступу

Файли довільного доступу зберігають дані в структурованому для звернення вигляді. В таких файлах можливий безпосередній доступ до даних за номером запису. Записуються дані у такі файли оператором `Put`, а читаються - оператором `Get`. В прикладі у файл довільного доступу записуються дані про студентів - значення змінних типу `student`. Після цього дані з файлу читаються і виводиться інформація про студентів з заданим прізвищем.

```
Type student 'тип користувача
    name As String * 20
    Ball As Double
End Type
Dim obj As student 'змінна типу student
Dim s1, s2 As String
```

```
'процедура main
Public Sub main()
'відкрити файл c:\file2.dat під номером 1 для довільного
доступу; довжина запису файлу = Len(obj)
Open "c:\file2.dat" For Random As #1 Len = Len(obj)
Do 'початок циклу
    s1 = InputBox("Введіть ім'я", "Ім'я") 'ввести ім'я
    If s1 = "" Then Exit Do 'якщо нічого не введено, то
вийти з циклу
    s2 = InputBox("Введіть бал", "Бал") 'ввести бал
    If s2 = "" Then Exit Do 'якщо нічого не введено, то
вийти з циклу
    obj.name = s1 'присвоїти obj.Name значення
    obj.Ball = CDbl(s2) 'присвоїти obj.Ball значення
    Put #1, , obj 'записати obj у поточну позицію файлу 1
    Debug.Print obj.name; obj.Ball 'вивести
Loop 'повторити
Close #1 'закрити файл 1

'відкрити файл c:\file2.dat під номером 1 для довільного
доступу
Open "c:\file2.dat" For Random As #1 Len = Len(obj)
Do While Not EOF(1) 'поки не кінець файлу 1
    Get #1, , obj 'прочитати дані з поточної позиції у obj
    'якщо знайдено ім'я "Ivanov" і не кінець файлу, то
    If Trim(obj.name) = "Ivanov" And Not EOF(1) Then _
        'вивести позицію запису і дані
        Debug.Print Seek(1) - 1; obj.name; obj.Ball
Loop 'повторити
Get #1, 1, obj 'прочитати дані з позиції 1
Debug.Print 1; obj.name; obj.Ball 'вивести дані
Close #1 'закрити файл
End Sub
```

Об'єктно-орієнтоване програмування

Об’єктно-орієнтоване програмування (ООП) основане на використанні об’єктів - абстрактних моделей реальних предметів чи понять [5, 19, 20, 23, 28, 29, 32]. ООП є близьким до природного способу мислення людини, характерне для багатьох високорівневих мов програмування та дозволяє суттєво спростити розробку складних програмних систем.

Об’єкти створюються за допомогою спеціальних типів даних - класів. Кожен клас описує множину об’єктів певного типу. Об’єкти можуть володіти властивостями, методами і подіями. Властивість (ще використовують терміни поле, атрибут-дане) - це описана в класі змінна або константа, яка визначає певну характеристику об’єкта (висота, ширина, колір тощо) і має певне значення. Наприклад код `object.color="red"` присвоює значення `"red"` властивості `color` об’єкта `object`. Метод - це описана в класі процедура, яка застосовує певний алгоритм до об’єкта. Наприклад код `object.draw` викликає метод `draw` об’єкта `object`. Подія - це дія, яка розпізнається програмою і обробляється процедурою обробки події. Подію може викликати сама програма, система або користувач.

В прикладі створюється об’єкт `obj` класу `Class1`, його властивостям `z`, `x` присвоюються значення і викликається метод `y`. Для запуску цього прикладу необхідно додатково створити модуль класу `Class1` (дивіться рис. 1.5 та приклад нижче).

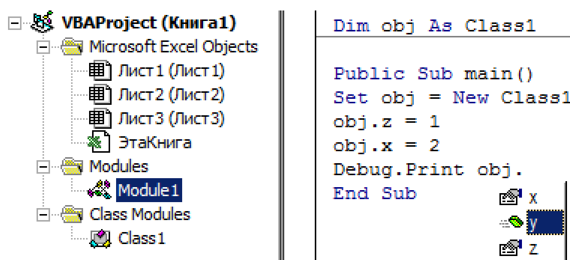


Рисунок 1.5 - Модулі проекту

```
Dim obj As Class1 'оголосити об'єкт класу Class1
```

```
Public Sub main()
Set obj = New Class1 'створити об'єкт obj
obj.z = 1 'присвоїти властивості 'z' значення
obj.x = 2 'присвоїти властивості 'x' значення
Debug.Print obj.y(5) 'викликати метод 'y' з параметром 5
End Sub
```

32

Модуль класу Class1

Клас – похідний тип даних, який описує множину об'єктів, в яких дані різного типу (властивості) об'єднані з алгоритмами їх опрацювання (методами). VBA-класи можуть також містити події. Наступний модуль описує клас `Class1`, який містить такі члени: змінні `arg` і `z`, подію `notNumber`, властивість `x`, метод-функцію `y` і процедуру ініціалізації `Class_Initialize`. Властивість `x` відрізняється від інших змінних класу тим, що процедура `Property Get x` автоматично викликається під час отримання значення властивості `x`, а процедура `Property Let x` автоматично викликається під час присвоєння значення властивості `x`. Про використання події `notNumber` дивіться приклад "Обробка подій користувача" нижче.

```
Dim arg As Variant 'поле класу (закрите)
Public z As Integer 'поле класу (загальнодоступне)
Public Event notNumber(x) ' подія (загальнодоступна)

Private Sub Class_Initialize() 'процедура ініціалізації
arg = 0
End Sub

'процедура повернення властивістю 'x' значення дійсного
типу
Public Property Get x() As Variant
x = arg 'присвоїти властивості значення
End Property
```

```
'процедура присвоєння властивості 'x' значення дійсного
типу
Public Property Let x(ByVal vNewValue As Variant)
If Not IsNumeric(vNewValue) Then 'якщо не числове дане, то
    RaiseEvent notNumber(vNewValue) ' викликати подію
    Exit Property ' вийти
End If
arg = vNewValue 'присвоїти arg нове значення
End Property

'метод класу (функція) 'y' з параметром 's'
Public Function y(s As Variant)
y = z * arg ^ s
End Function
```

РОЗДІЛ 2. ЕЛЕМЕНТИ КЕРУВАННЯ ДЛЯ ПРОГРАМ З GUI

Клас UserForm - форма користувача

Елемент керування ActiveX (OLE) – це внутрішньопроцесний сервер COM у вигляді бінарного програмного модуля (DLL), який являє собою універсальний програмний компонент, що може інтегруватися в програми (клієнти COM) для розширення їх можливостей. Елемент ActiveX володіє властивостями, методами і подіями. Бібліотека Microsoft Forms 2.0 Object Library містить ActiveX-компоненти (елементи керування) для створення графічного інтерфейсу користувачу (GUI) програми [23, 28-30, 32, 41]. Щоб створити програму з GUI потрібно додати новий модуль шляхом вибору меню Insert/UserForm. З'явиться новий модуль `UserForm1` і заготовка вікна (форми) `UserForm1`, в яку можна перетягнути потрібні елементи керування з вікна Toolbox (рис. 2.1). В вікно Toolbox також можна додати елементи керування з інших бібліотек шляхом вибору меню Additional Controls...

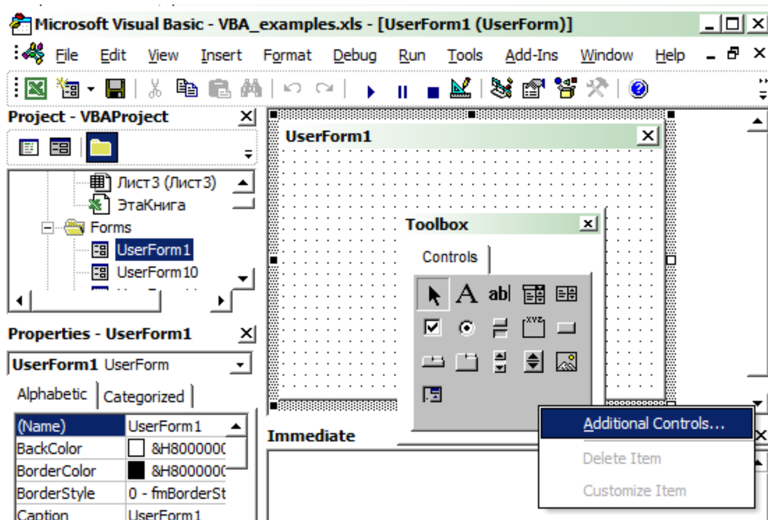


Рисунок 2.1 - Проектування GUI в редакторі Visual Basic

Класи бібліотеки MS Forms можна переглянути в Object Browser (рис. 2.2). Класи містять властивості (Property), методи (Method) і події (Event).

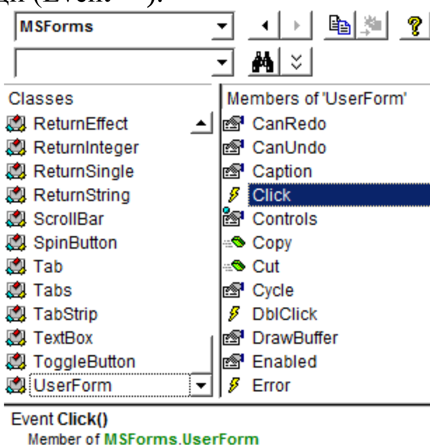


Рисунок 2.2 - Перегляд бібліотеки MS Forms 2.0 в вікні Object Browser

Значення властивостей вибраного об’єкта можна змінювати на етапі проектування у вікні Properties (рис. 2.3). За допомогою подій компоненти можуть реагувати на дії користувача. Коли відбувається подія компонента, то автоматично виконується процедура обробки цієї події. Щоб пов’язати задану подію з процедурою обробки події, потрібно в списках, які знаходяться відразу над кодом програми в редакторі Visual Basic, вибрати компонент і відповідну його подію (рис. 2.3). Автоматично в код буде додана пуста процедура, що оброблятиме цю подію. Тепер в цю процедуру можна додати який-небудь код.

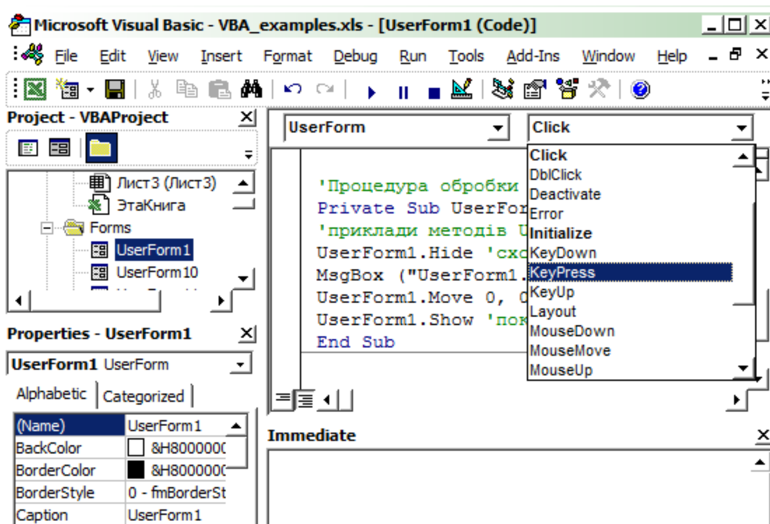


Рисунок 2.3 - Властивості і події в редакторі Visual Basic

Клас `UserForm` описує поняття вікна програми (форми користувача). В наступному прикладі показано використання основних властивостей, методів і подій цього класу. Процедура

UserForm_Initialize обробляє подію Initialize (ініціалізація форми), процедура UserForm_Click - подію Click (натиск лівої кнопки миші на формі), процедура UserForm_Terminate - подію Terminate (знищення форми). Результат роботи програми показаний на рис. 2.4.

```
'Процедура обробки події Initialize (ініціалізація форми)
Private Sub UserForm_Initialize()
'приклади властивостей UserForm:
UserForm1.Caption = "Перша форма" 'надпис
UserForm1.BackColor = vbGreen 'колір фону
UserForm1.BorderStyle = fmBorderStyleSingle 'стиль границі
UserForm1.Picture = LoadPicture("C:\WINDOWS\Паркет.bmp")
'фонова картинка
UserForm1.PictureSizeMode = fmPictureSizeModeStretch
'розмір картинки
UserForm1.StartUpPosition = 0 'початкова позиція
'координати верхнього лівого кута:
UserForm1.Left = 50
UserForm1.Top = 50
UserForm1.Height = 100 'висота
UserForm1.Width = 200 'ширина
UserForm1.MousePointer = fmMousePointerCross 'вид
вказівника миші
UserForm1.Enabled = True 'чи допустиме керування вручну
'UserForm1.ShowModal = False 'зробити не модальною.
Властивість змінюється тільки на етапі проектування!
End Sub

'Процедура обробки події Click (натиск лівої кнопки миші)
Private Sub UserForm_Click()
'приклади методів UserForm:
UserForm1.Hide 'сховати
MsgBox ("UserForm1.Show") 'вивести вікно з повідомленням
UserForm1.Move 0, 0, 300, 400 'перемістити і змінити
розмір
UserForm1.Show 'показати
```

```
End Sub

'Процедура обробки події Terminate (знищення)
Private Sub UserForm_Terminate()
MsgBox ("Відбулась подія Terminate") 'вивести вікно з
повідомленням
End Sub
```

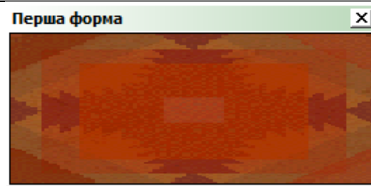


Рисунок 2.4 - Результат роботи програми

Модуль Module1

Можливо також створити додаткові модулі, з яких можна звертатись до модуля `UserForm1`. Нижче показано модуль `Module1` з процедурою `main`, в якій викликається метод `Show` об'єкта `UserForm1`. Якщо виконати цю процедуру, то вікно `UserForm1` з'явиться на екрані.

```
Public Sub main()
UserForm1.Show
End Sub
```

Обробка подій користувача

Об'єкт оголошений з ключовим словом `WithEvents` може викликати події, які описані в його класі. В наступному прикладі таким об'єктом є `obj` класу `Class1` (дивіться модуль класу `Class1`). В класі `Class1` подія `notNumber` викликається оператором `RaiseEvent` тоді, коли змінна `vNewValue` не є числовим даним. Для створення процедури-обробника `obj_notNumber` виберіть об'єкт `obj` і подію `notNumber` у списках

над кодом програми. Подія буде виникати після введення в поле `TextBox1` нечислового значення і оброблятись процедурою `obj_notNumber` (рис. 2.5).

```
'Оголосити об'єкт класу Class1
Dim WithEvents obj As Class1 ' об'єкт може викликати події

Private Sub CommandButton1_Click()
Set obj = New Class1 'створити об'єкт obj
obj.z = 1
obj.x = TextBox1.Text 'тут може виникнути подія notNumber
TextBox1.Text = obj.y(5) 'вивести результат в TextBox1
End Sub

'Обробник події користувача notNumber
Private Sub obj_notNumber(x As Variant)
MsgBox "Warning! x is not number: " & x
End Sub
```

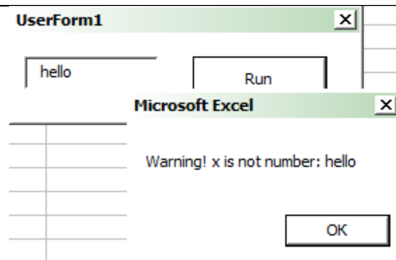


Рисунок 2.5 - Обробка події notNumber

Загальні властивості компонентів MS Forms

В наступному прикладі `UserForm2` показано використання методів і властивостей, які є у більшості елементів керування MS Forms. Доступ до усіх елементів керування форми можна отримати за допомогою властивості `Controls` (рис. 2.6).

```
Private Sub UserForm_Initialize()
```

```
'для кожного елемента керування на UserForm2
For Each obj In UserForm2.Controls
    obj.AutoSize = True 'авторозмір
    obj.Visible = True 'видимість
    obj.Enabled = True 'дозвіл керування
    'координати верхнього лівого кута
    obj.Left = 10
    obj.Top = obj.Top + 20
    obj.Height = 20 'висота
    obj.Width = 100 'ширина
    obj.ControlTipText = "help" 'текст підказки
    obj.BackColor = vbYellow 'колір фону
    obj.ForeColor = RGB(0, 0, 0) 'колір переднього плану
    obj.BackStyle = fmBackStyleTransparent 'тип фону
Next obj
'метод SetFocus
CommandButton1.SetFocus 'установити фокус на кнопці
Debug.Print UserForm2.ActiveControl.Value 'значення
активного елемента, який містить фокус
End Sub
```

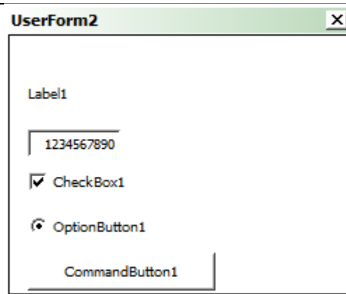


Рисунок 2.6 - Результат роботи програми

Клас Label - надпис

В прикладі `UserForm3` показано використання елемента керування `Label` (надпис), який використовується для

відображення тексту на вікні (рис. 2.7). Основною властивістю є `Caption`, яка містить рядок з текстом.

```
'Процедура обробки події Click (натиск лівої кнопки миші)
Private Sub Label1_Click()
Label1.AutoSize = False 'авторозмір
Label1.Height = 40 'висота
Label1.Width = 70 'ширина
Label1.Font.name = "Times New Roman" 'ім'я шрифту
Label1.Font.Size = 14 'розмір шрифту
Label1.Font.Bold = True 'жирний шрифт
Label1.Font.Italic = True 'курсив шрифт
Label1.Font.Underline = True 'підкреслений шрифт
Label1.ForeColor = RGB(255, 0, 0) 'колір переднього плану
(шрифту)
Label1.TextAlign = fmTextAlignCenter 'вирівнювання надпису
Label1.SpecialEffect = fmSpecialEffectSunken 'спеціальний
візуальний ефект
Label1.Caption = "Clicked" 'надпис
Label1.WordWrap = True 'перенос тексту
End Sub
```

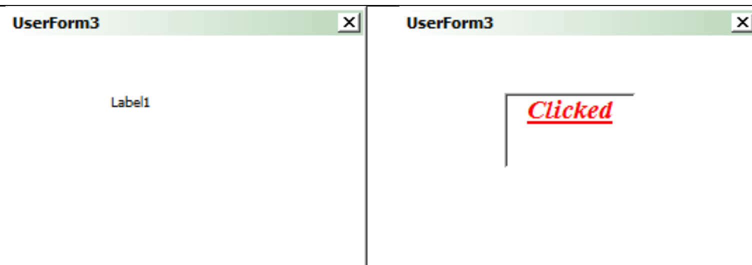


Рисунок 2.7 - Елемент керування Label

Клас TextBox - текстове поле

В прикладі `UserForm4` показано використання елемента керування `TextBox` (текстове поле), який використовується для

введення і виведення тексту (рис. 2.8). Основною властивістю є Text, яка містить рядок з текстом.

```
Dim x As Double
Dim s As String

'Процедура обробки події Initialize (ініціалізація форми)
Private Sub UserForm_Initialize()
TextBox1.AutoSize = False 'авторозмір
TextBox1.TextAlign = fmTextAlignCenter 'вирівнювання
тексту
TextBox1.Enabled = False 'доступність
TextBox1.Text = "Hello"  'текст
TextBox1.Text = 5.71 'текст
x = CDbl(TextBox1.Text) 'конвертувати текст в дійсне число
'або
x = CDbl(TextBox1.Value) 'значення
TextBox1.SelStart = 0 'початкова позиція виділення
TextBox1.SelLength = TextBox1.TextLength 'довжина
виділення
TextBox1.Copy 'скопіювати в буфер обміну
TextBox2.MaxLength = 8 'максимальна довжина тексту
TextBox2.PasswordChar = "*"  'символ для введення пароля
TextBox3.MultiLine = True 'багаторядковий режим
TextBox3.Height = 50 'висота
TextBox3.Font.Size = 12 'розмір шрифту
TextBox3.ScrollBars = fmScrollBarsBoth 'смуги
прокручування
TextBox3.TabIndex = 2 'порядок зміни фокусу клавішею Tab
TextBox3.TabKeyBehavior = True 'дозволити вводити у текст
табуляцію клавішею Tab
'присвоїти текст у двох рядках
TextBox3.Text = "Перший рядок" & Chr(13) & "Другий рядок"
TextBox3.SetFocus 'установити фокус
End Sub

'процедура обробки події MouseUp (відпущено кнопку миші)
```

```
Private Sub TextBox3_MouseUp(ByVal Button As Integer,
ByVal Shift As Integer, ByVal x As Single, ByVal y As
Single)
s = TextBox3.SelText 'присвоїти виділений текст
End Sub
'процедура обробки події MouseDown (натиснуто кнопку миші)
Private Sub TextBox4_MouseDown(ByVal Button As Integer,
ByVal Shift As Integer, ByVal x As Single, ByVal y As
Single)
TextBox4.Text = s 'присвоїти текст
End Sub

'процедура обробки події Change (зміна значення)
Private Sub TextBox4_Change()
TextBox1.Text = TextBox4.Text 'присвоїти текст
End Sub

'процедура обробки події KeyPress (натиснута клавіша на
клавіатурі)
Private Sub TextBox4_KeyPress(ByVal KeyAscii As
MSForms.ReturnInteger)
'якщо ASCII код клавіші <48 або >57 (не цифра)
If KeyAscii < 48 Or KeyAscii > 57 Then
KeyAscii = 0 'не виводити нічого
End If
End Sub
```

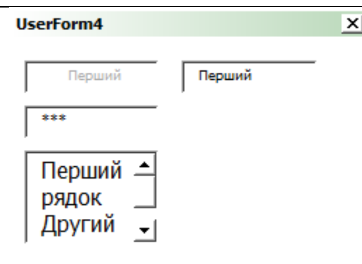


Рисунок 2.8 - Елемент керування TextBox

Клас CommandButton - кнопка

В прикладі `UserForm5` показано використання елемента керування `CommandButton` (кнопка), на який користувач може натиснути (рис. 2.9). Основною подією є `Click`, яка викликається натиском лівої кнопки миші на елементі керування.

```
'Процедура обробки події Click (натиск лівої кнопки миші)
Private Sub CommandButton1_Click()
CommandButton1.Enabled = True 'доступність
CommandButton1.Locked = False 'заблокованість
CommandButton1.Caption = "Click!" 'надпис
CommandButton1.AutoSize = True 'авторозмір
CommandButton1.Cancel = True 'асоціація з клавішею Esc
CommandButton1.Default = True 'асоціація з клавішею Enter
CommandButton1.Accelerator="A" 'клавіша-акселератор Alt-A
'фонова картинка
CommandButton1.Picture =
LoadPicture("d:\WINDOWS\Паркет.bmp")
'позиція картинки
CommandButton1.PicturePosition =
fmPicturePositionAboveLeft
End Sub

'Процедура обробки події DblClick (подвійний натиск лівої
кнопки миші)
Private Sub CommandButton1_DblClick(ByVal Cancel As
MSForms.ReturnBoolean)
CommandButton1.Caption = "DblClick!" 'надпис
End Sub
```

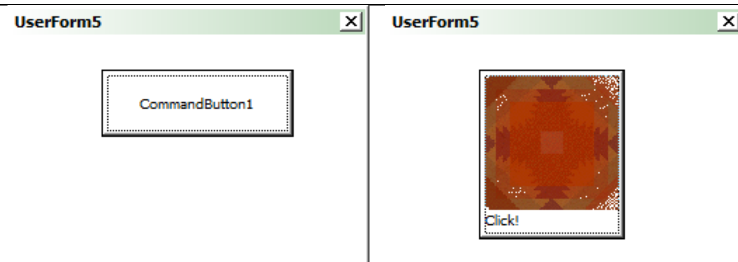


Рисунок 2.9 - Елемент керування CommandButton

Класи CheckBox і ToggleButton - прапорець і вимикач

В прикладі `UserForm6` показано використання елементів керування `CheckBox` (прапорець) і `ToggleButton` (вимикач), які володіють властивістю `Value`, що має значення `True`, коли стан елемента керування вибраний, і `False`, коли стан не вибраний. (рис. 2.10).

```
Private Sub UserForm_Initialize()
CheckBox1.Enabled = True 'доступність
CheckBox1.TripleState = True 'дозволити три стани
CheckBox1.Value = True 'значення "вибрано"
CheckBox1.Value = False 'значення "не вибрано"
CheckBox1.Value = Null 'значення "третій стан"
CheckBox1.SetFocus 'установити фокус
ToggleButton1.Value = False 'значення "не вибрано"
End Sub

'Процедура обробки події Change (зміна стану)
Private Sub CheckBox1_Change()
If CheckBox1.Value Then 'якщо значення=True
    CheckBox1.Caption = "True" 'змінити надпис на "True"
ElseIf Not CheckBox1.Value Then 'інакше, якщо
значення=False
    CheckBox1.Caption = "False" 'змінити надпис на "False"
Else: CheckBox1.Caption = "Null" 'інакше змінити надпис на
"Null"
End If
End Sub
```

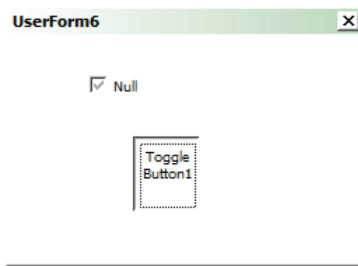


Рисунок 2.10 - Елементи керування CheckBox і ToggleButton

Клас OptionButton - перемикач

В прикладі UserForm7 показано використання елемента керування OptionButton (перемикач), який володіє властивістю Value, що має значення True, коли елемент керування вибраний, і False, коли він не вибраний (рис. 2.11).

```
Private Sub UserForm_Initialize()
OptionButton3.Value = True 'значення
OptionButton3.Caption = "True"  'надпис
End Sub

'Процедура обробки події Click (натиск лівої кнопки миші)
Private Sub CommandButton1_Click()
'надпис рамки змінити на ім'я активного елемента
Frame1.Caption = Frame1.ActiveControl.name
For Each opt In Frame1.Controls 'для кожного елемента
    opt.Caption = opt.Value 'змінити надпис на його
значення
Next opt
End Sub
```

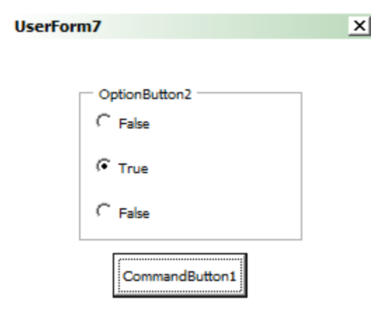


Рисунок 2.11 - Елемент керування OptionButton

Клас ListBox - список

В прикладі `UserForm8` показано використання елемента керування `ListBox` (список), який використовується для вибору елемента зі списку (рис. 2.12). Основними властивостями є `ListIndex` (індекс вибраного елемента) і `ListText` (текст вибраного елемента).

```
Private Sub UserForm_Initialize()
ListBox1.ListStyle = fmListStyleOption 'стиль списку
ListBox1.TextAlign = fmTextAlignCenter 'вирівнювання
тексту
ListBox1.MatchEntry = fmMatchEntryFirstLetter 'пошук по
першій букві
'перший спосіб заповнення списку:
ListBox1.AddItem "Перший"  'перший елемент списку
ListBox1.AddItem "Другий"  'другий елемент списку
ListBox1.AddItem "Третій"  'третій елемент списку
ListBox1.Clear 'очистити список
'другий спосіб заповнення списку:
ListBox1.List = Array("Перший", "Другий", "Третій")
End Sub

'процедура обробки події Click (натиск лівою кнопкою миші)
Private Sub ListBox1_Click()
```

```
'вивести в надпис форми вибраний елемент, його індекс,
кількість елементів
UserForm8.Caption = ListBox1.Text & " " &
ListBox1.ListIndex _
& "/" & ListBox1.ListCount
End Sub

'процедура обробки події KeyDown (опущена клавіша на
клавіатурі)
Private Sub ListBox1_KeyDown(ByVal KeyCode As
MSForms.ReturnInteger, ByVal Shift As Integer)
'якщо натиснуто "Insert", додати новий елемент
If KeyCode = vbKeyInsert Then ListBox1.AddItem "Новий",
ListBox1.ListIndex + 1
'якщо натиснуто "Delete", видалити елемент
If KeyCode = vbKeyDelete Then ListBox1.RemoveItem
ListBox1.ListIndex
End Sub
```

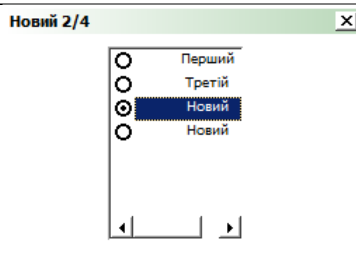


Рисунок 2.12 - Елемент керування ListBox

ListBox з кількома колонками і мультивибір

В прикладі `UserForm9` показано можливість створення елементів керування `ListBox` з кількома колонками та можливість вибору кількох елементів з одного списку (рис. 2.13).

```
Private Sub UserForm_Initialize()
Dim A(0 To 1, 0 To 1) As String 'масив
```

```
'заповнити масив
A(0, 0) = 1
A(0, 1) = "Перший"
A(1, 0) = 2
A(1, 1) = "Другий"
ListBox1.ColumnCount = 2 'кількість колонок
ListBox1.List = A 'заповнити список масивом
ListBox1.ColumnWidths = "20;20" 'ширина колонок
ListBox1.TextColumn = 2 'колонка, елемент якої
повертається Text
End Sub

'процедура обробки події KeyDown (опущена клавіша на
клавіатурі)
Private Sub ListBox1_KeyDown(ByVal KeyCode As
MSForms.ReturnInteger, ByVal Shift As Integer)
'якщо натиснуто "Insert"
If KeyCode = vbKeyInsert Then
    'якщо виключений мультивибір, то включити, і навпаки
    ListBox1.MultiSelect = IIf(ListBox1.MultiSelect =
fmMultiSelectSingle, fmMultiSelectMulti,
fmMultiSelectSingle)
End If
'якщо натиснуто "Delete"
If KeyCode = vbKeyDelete Then
    'i змінюється від 0 до кількості елементів-1
    For i = 0 To ListBox1.ListCount - 1
        'якщо елемент вибраний, додати його в ListBox2
        If ListBox1.Selected(i) Then ListBox2.AddItem
ListBox1.List(i, 1)
    Next i
End If
End Sub
```

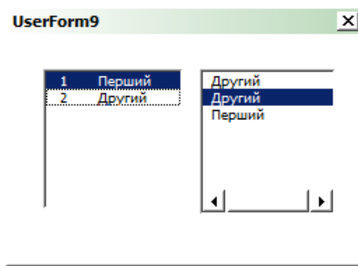


Рисунок 2.13 - ListBox з кількома колонками і мультивибір

Клас ComboBox - список з текстовим полем

В прикладі UserForm10 показано використання елемента керування ComboBox (комбінований список), який являє собою поєднання TexBox і ListBox (рис. 2.14). Має ті ж властивості, що ListBox і додатково має властивість Text (рядок в текстовому полі).

```
Private Sub UserForm_Initialize()
ComboBox1.List = Array(1, 2, 3, 4, 5, 6, 7) 'заповнити
список
ComboBox1.ListRows = 4 'у списку показувати 4 рядка
ComboBox1.MatchRequired = True 'заборона введення у
текстове поле значень, яких немає у списку
End Sub

'процедура обробки події Change (зміна значення текстового
поля)
Private Sub ComboBox1_Change()
'вивести у надпис форми текст текстового поля і індекс
вибраного елемента
UserForm10.Caption = ComboBox1.Text & " " &
ComboBox1.ListIndex
End Sub
```

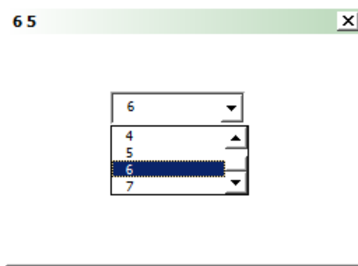


Рисунок 2.14 - Елемент керування ComboBox

Класи ScrollBar і SpinButton - смуга прокручування і лічильник

В прикладі UserForm11 показано використання елементів керування ScrollBar (смуга прокручування) і SpinButton (лічильник), які дозволяють користувачу виставити задане значення їх властивості Value шляхом натиску кнопок зі стрілками або переміщення повзунка (рис. 2.15). Основна їх подія - Change (зміна значення).

```
Private Sub UserForm_Initialize()
ScrollBar1.Orientation = fmOrientationHorizontal
'орієнтація
ScrollBar1.Min = 0 'мінімальне значення
ScrollBar1.Max = 10 'максимальне значення
ScrollBar1.SmallChange = 1 'малий крок зміни значення
ScrollBar1.LargeChange = 2 'великий крок зміни значення
ScrollBar1.Delay = 10 'затримка подій зміни значення
SpinButton1.Min = 0 'мінімальне значення
SpinButton1.Max = 10 'максимальне значення
SpinButton1.SmallChange = 1 'малий крок зміни значення
End Sub

'процедура обробки події Change (зміна значення)
Private Sub ScrollBar1_Change()
'змінити надпис на формі на значення ScrollBar1
UserForm11.Caption = ScrollBar1.Value
```

```
End Sub

'процедура обробки події SpinUp (натиснуто кнопку "вверх")
Private Sub SpinButton1_SpinUp()
'збільшити значення на 2
SpinButton1.Value = SpinButton1.Value + 2
End Sub

'процедура обробки події Change (зміна значення)
Private Sub SpinButton1_Change()
'змінити надпис на формі на значення SpinButton1
UserForm11.Caption = SpinButton1.Value
End Sub
```

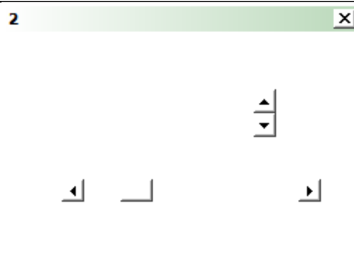


Рисунок 2.15 - Елементи керування `ScrollBar` і `SpinButton`

Клас TabStrip - набір вкладок

В прикладі `UserForm12` показано використання елемента керування `TabStrip` (набір вкладок), за допомогою якого користувач може вибрати потрібну вкладку (рис. 2.16). На `TabStrip` можна розташувати інші елементи керування, але одні і ті ж для різних вкладок. Основною властивістю є `SelectedItem.Index` (індекс вибраної вкладки), а основна подія - `Change` (зміна вкладки).

```
Private Sub UserForm_Initialize()
TabStrip1.MultiRow = True 'дозволити кілька рядків вкладок
TabStrip1.TabOrientation = fmTabOrientationLeft
```

```
'орієнтація
TabStrip1.TabIndex = 1 'вибрано вкладку з індексом 1
TabStrip1.Tabs.Item(0).Caption = "A" 'надпис першої
вкладки
TabStrip1.Tabs.Item(1).Caption = "B" 'надпис другої
вкладки
TabStrip1.Tabs.Add "Tab3", "C", 2 'додати третю вкладку
End Sub

'процедура обробки події Change (зміна вкладки)
Private Sub TabStrip1_Change()
'змінити надпис на індекс вибраної вкладки
Label1.Caption = TabStrip1.SelectedItem.Index
End Sub
```

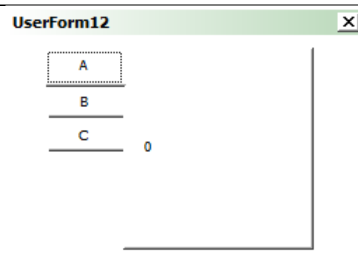


Рисунок 2.16 - Елемент керування TabStrip

Клас MultiPage - набір сторінок

В прикладі UserForm13 показано використання елемента керування MultiPage (набір сторінок), за допомогою якого користувач може вибрати потрібну сторінку з розташованими тільки на ній елементами керування (рис. 2.17). Основною властивістю є SelectedItem.Index (індекс вибраної сторінки), а основна подія - Change (зміна сторінки).

```
Dim TextBox1 As Control 'об'єкт TextBox1

Private Sub UserForm_Initialize()
```

```
MultiPage1.Pages(0).Caption = "A" 'змінити надпис сторінки
MultiPage1.Pages(1).Caption = "B" 'змінити надпис сторінки
'створити новий об'єкт TextBox1
Set TextBox1 =
MultiPage1.Pages(1).Controls.Add("Forms.TextBox.1",
"TextBox1", Visible)
TextBox1.Visible = True 'зробити видимим
End Sub

'процедура обробки події Click
Private Sub CommandButton1_Click()
TextBox1.Text = "Hello!" 'змінити текст
End Sub

'процедура обробки події Change (сторінка змінена)
Private Sub MultiPage1_Change()
'змінити надпис форми на індекс вибраної сторінки
UserForm13.Caption = MultiPage1.SelectedItem.Index
End Sub
```

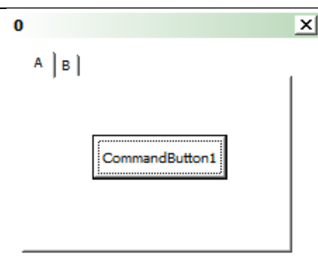


Рисунок 2.17 - Елемент керування MultiPage

Клас Image - картинка

В прикладі `UserForm14` показано використання елемента керування `Image` (картинка), яка призначена для виведення довільного растрового зображення на форму (рис. 2.18). Основна властивість - `Picture`, якій слід присвоїти результат функції `LoadPicture`, що завантажує растрове зображення з файлу.

```
Private Sub UserForm_Initialize()
'з об'єктом Image1
With Image1
    .Picture = LoadPicture("d:\WINDOWS\Паркет.bmp")
'завантажити картинку
    .PictureSizeMode = fmPictureSizeModeClip 'розмір
картинки
    .PictureTiling = True 'замостити
    .PictureAlignment = fmPictureAlignmentCenter
'вирівнювання
End With
End Sub
```



Рисунок 2.18 - Елемент керування Image

РОЗДІЛ 3. БІБЛІОТЕКИ КОМПОНЕНТІВ

Клас Application - програма Excel

Об’єктна модель Excel (Microsoft Excel 11.0 Object Library) дозволяє звертатись з VBA-програми до усіх основних компонентів програми Excel: робочих книг, робочих листів, комірок, діаграм [4, 6, 11, 15, 18, 19, 23, 29, 31-34, 40, 41, 43-46]. В прикладі показано використання основних властивостей і методів об’єкта `Application`, який являє собою саму програму Excel. Скопіюйте

наступний код в модуль `Module1` і створіть модуль `Module2` з пустою процедурою `main`.

```
Public Sub main()
'деякі властивості Application:
Debug.Print Application.ActiveWorkbook.Name 'ім'я активної книги, або скорочено ActiveWorkbook.Name
Debug.Print Application.ActiveSheet.Name 'ім'я активного листа
Debug.Print Application.ActiveCell.Value 'значення активної комірки
'ім'я активного листа книги, в якій виконується макрос
Debug.Print ThisWorkbook.ActiveSheet.Name
Application.Calculation = xlCalculationAutomatic 'режим обчислень
Application.Caption = "Моя програма" 'надпис
Application.Caption = Empty 'надпис за замовчуванням
Application.Cells(1, 1).Value = 1 'значення комірки Excel (1,1)
Application.DisplayStatusBar = True 'показувати рядок стану
Application.StatusBar = "Hello!" 'текст у рядку стану
Application.ScreenUpdating = False 'не оновлювати екран
Application.ScreenUpdating = True 'оновлювати екран
Debug.Print Application.Version 'версія Excel
Application.EnableCancelKey = xlInterrupt 'переривати виконання натиском Ctrl-Break
Application.WindowState = xlMaximized 'стан вікна

'деякі методи Application:
Application.Calculate 'обчислити книгу
'вивести діалогове вікно, результат присвоїти i
i = InputBox("Виконати (1),відкласти виконання (2),вийти (3)", "Вибір", 0)
Select Case i 'вибір i
Case 1
Application.Run "VBAProject.Module1.main" 'виконати макрос
```

```
Case 2
'відкласти виконання макроса на 10 секунд
Application.OnTime Now + TimeValue("0:00:10"),
"Module1.main"
Case 3
Application.Quit 'вийти з Excel
Case Else
End Select
Application.OnKey "^{a}", "Module1.main" 'виконати макрос
після натиску Ctrl-A
'конвертувати формулу з формату R1C1 у формат A1
Debug.Print Application.ConvertFormula("=SUM(R1C1:R5C1)",
xlR1C1, xlA1)
'виділити перетин діапазонів
Application.Intersect(Range("A1:B2"),
Range("B2:C3")).Select
'виділити об'єднання діапазонів
Application.Union(Range("A1:B2"), Range("B2:C3")).Select
answ = Application.Dialogs(xlDialogOpen).Show 'показати
діалогове вікно відкриття файлу

'Evaluate конвертує ім'я Excel в об'єкт або значення
Application.Evaluate("A2").Value = 2 'перетворити рядок в
об'єкт
[A2].Value = 2 ' або
Debug.Print Application.Evaluate("SUM(A1:A2)") '
перетворити рядок в функцію
'Debug.Print Application.Evaluate("SUM(1,2)") ' або
Set r = Range("A1:A2")
Debug.Print Application.Sum(r) ' виклик вбудованої функції
листа Excel
'Debug.Print Application.WorksheetFunction.Sum(r) ' або
'Debug.Print Application.Sum(1, 2) ' або

'Debug.Print my_funct(3) 'виклик функції користувача листа
Excel
End Sub
```

```
Книга1
Лист1

Лист1
11.0
=SUM($A$1:$A$5)
 3
 3
```

Запис макросів в Excel

Користувач може записати свої дії з програмою Excel у вигляді макроса. Цей макрос потім можна зберегти, виконувати і редагувати. За допомогою макросів можна вивчати об'єктну модель Excel навіть без перегляду документації. Виберіть в меню Excel Сервис/Макрос/Начать запись... Введіть назву макроса і натисніть Ok. Введіть в комірку `A1` значення 1, в комірку `A2` значення 2, в комірку `A3` формулу `=СУММ(A1:A2)`. Натисніть кнопку ■, щоб зупинити запис. Очистіть лист. Виконайте макрос з меню Сервис/Макрос/Макросы.../Выполнить. Макрос повинен відтворити ваші дії. Щоб відредагувати макрос натисніть Сервис/Макрос/Макросы.../Изменить.

```
Sub Макрос1()
    ActiveCell.FormulaR1C1 = "1"
    Range("A2").Select
    ActiveCell.FormulaR1C1 = "2"
    Range("A3").Select
    ActiveCell.FormulaR1C1 = "=SUM(R[-2]C:R[-1]C)"
    Range("A4").Select
End Sub
```

Функція користувача Excel

В прикладі показано створення функцію користувача Excel мовою VBA, яку можна буде використовувати в комірках листа

Excel. Створіть новий модуль, який містить код функції my_funct. В комірку Excel введіть формулу =my_funct(2) і натисніть Enter. Повинен з'явитись результат 4.

```
Public Function my_funct(x As Double)
my_funct = x + 2
End Function
```

Клас Range - діапазон комірок Excel

Клас Range описує окрему комірку Excel або діапазони комірок. Об'єкт цього класу можна отримати так: Range("<назва діапазону>"). В прикладі показані основні властивості і методи класу Range.

```
Public Sub main()
ActiveSheet.UsedRange.Clear 'очистити діапазон з
значеннями
ActiveSheet.UsedRange.ClearContents 'очистити вміст
ActiveSheet.UsedRange.ClearComments 'очистити коментарі
ActiveSheet.UsedRange.ClearFormats 'очистити формати
'змінити значення комірки A1 листа Лист1 книги
VBA_examples
Workbooks("VBA_examples").Worksheets("Лист1").Range("A1").
Value = 5
Range("Лист1!A1").Value = 5 'або так
Range("A1").Value = 5 'або так
Cells(1, 1).Value = 5 'або так
Rows(10).RowHeight = StandardHeight 'стандартна висота 10-
го рядка
Columns(10).ColumnWidth = StandardWidth 'стандартна ширина
10-го стовпця
Range("E5").EntireRow.Clear 'очистити рядок 5
Range("E5").EntireColumn.Clear 'очистити стовпчик E
Range("A1").AddComment "Коментар" 'додати коментар
Debug.Print Range("A1").Comment.Text 'текст коментаря
```

```
Range("A1").Font.Size = 12 'розмір шрифту
Range("A1").Font.Color = RGB(255, 0, 0) 'колір шрифту
Range("A1:B2").name = "MyRange" 'назва діапазону
Debug.Print Range("A1:B2").Count 'кількість комірок
Range("B1:B2").Formula = "=$A$1+1" 'формула
Range("B1:B2").FormulaR1C1 = "=(R1C1)+1" 'формула в
форматі R1C1
Range("C1:D2").FormulaArray = "=TRANSPOSE(A1:B2)" 'формула
масиву
Range("E1").FormulaLocal = "=СУММ(C1:D2)" 'формула
неангломовної версії Excel
Debug.Print Range("E2").Text 'вміст у текстовому форматі
Debug.Print Range("A1").Address(True, False) 'адреса
Debug.Print Range("C1:D2").Rows.Count 'кількість рядків
Debug.Print Range("C1:D2").Columns.Count 'кількість
стовпців
Range("A1").EntireColumn.AutoFit 'авторозмір
Range("A3:A4").Cut 'вирізати в буфер обміну
Range("A1").Copy 'скопіювати в буфер обміну
Range("A4").PasteSpecial xlPasteValues 'вставити з буфера
значення
Range("H10").Delete 'знищити
Rows(1).Insert 'вставити новий рядок перед першим рядком
Range("A1").Offset(2, 0).Value = 1 'змінити значення
комірки А3
Range("A4:A5").Select 'виділити діапазон
Selection.Copy 'копіювати виділення в буфер обміну

'методи для роботи з діапазонами
'заголовки полів
Range("F1").Value = "Значення": Range("G1").Value =
"Номер"
Range("F2").Value = 1 'перше значення
'геометрична прогресія xlGrowth з кроком 2 і кінцевим
значенням 16
Range("F2").DataSeries xlColumns, xlGrowth, Step:=2,
Stop:=16
```

```
'перші значення
Range("G2").Value = 1
Range("G3").Value = 2
Range("G2:G3").AutoFill Range("G2:G6") 'автозаповнення:
1,2,3,4,5
'автофільтр по другому стовпчику, значення 2 або 3
Range("F2:G6").AutoFilter 2, "2", xlOr, "3"
Range("F2:G6").AutoFilter 'відмінити автофільтр
'заголовки полів для критеріїв
Range("F7").Value = "Значення": Range("G7").Value =
"Номер"
Range("G8").Value = 4 'критерій
'розширений фільтр копіює знайдене (за критерієм F7:G8) в
діапазон H1:I1
Range("F1:G6").AdvancedFilter xlFilterCopy,
Range("F7:G8"), Range("H1:I1")
Range("F2:G6").Sort Range("G2"), xlDescending 'сортування
за спаданням
Range("G2:G6").Find("4").Activate 'активувати комірку зі
знайденим значенням 4
End Sub
```

Класи Worksheet і Worksheets - робочий лист і листи Excel

Класи `Worksheet` і `Worksheets` описують робочий лист і листи Excel. Об’єкт листа можна отримати так: `Worksheets(<індекс або назва листа>)`. В прикладі показані основні властивості і методи цих класів.

```
Public Sub main()
Worksheets(1).Activate 'активувати перший лист
ActiveSheet.Visible = True 'зробити видимим активний лист
ActiveSheet.UsedRange.Clear 'очистити діапазон з
значеннями
Debug.Print ActiveSheet.StandardHeight 'стандартна висота
рядків
Debug.Print Worksheets.Count 'кількість листів
```

```
Worksheets.Add 'додати лист на початок
s = Worksheets(1).name 'присвоїти ім'я першого листа
Worksheets(1).Copy Worksheets(3) 'скопіювати перед третім
листом
Worksheets(1).Move Worksheets(3) 'перемістити перед третім
листом
Worksheets(s).Delete 'знищити
Worksheets(2).Delete 'знищити
Worksheets(2).Activate 'активувати другий лист
Worksheets("Лист1").Activate 'активувати лист "Лист1"
End Sub
```

Класи Workbook і Workbooks - робоча книга і книги Excel

Класи `Workbook` і `Workbooks` описують поняття робочої книги і усіх відкритих книг Excel. Об’єкт книги можна отримати так: `Workbooks(<індекс або назва книги>)`. В прикладі показані основні властивості і методи цих класів.

```
Public Sub main()
Debug.Print ActiveWorkbook.ActiveSheet.name 'ім'я
активного листа активної книги
Debug.Print ActiveWorkbook.Worksheets(1).name 'ім'я
першого листа активної книги
Debug.Print ActiveWorkbook.Saved 'чи збережено
Debug.Print Workbooks.Count 'кількість робочих книг у
сімействі Workbooks
Workbooks.Add 'додати книгу
Workbooks(2).Activate 'активувати другу книгу
Workbooks(2).Password = "Пароль" 'установити пароль на
другу книгу
Workbooks(2).Password = "" 'зняти пароль
Workbooks(2).SaveAs "my_book2" 'зберегти як my_book2.xls
Workbooks(2).Save 'зберегти
Debug.Print Workbooks(2).HasPassword 'чи має пароль
Workbooks(2).PrintOut 'вивести на друк
Workbooks(2).Close 'закрити
```

```
Workbooks.Open "my_book2" 'відкрити my_book2.xls
'змінити значення комірки A1 листа Лист1 книги my_book2
Workbooks("my_book2").Worksheets("Лист1").Range("A1").Valu
e = 3
Workbooks("my_book2").Close 'закрити my_book2.xls
Kill "my_book2.xls" 'знищити файл my_book2.xls
End Sub
```

Події об'єкта Workbook

В прикладі показано обробку таких подій робочої книги Excel, як `Open`, `BeforeClose`, `SheetSelectionChange`, `SheetChange`, `SheetBeforeRightClick`. Вставте наступний код в об'єктний модуль `ЭтаКнига` класу `Workbook`. В вікні Immediate будуть виводитись повідомлення, що відповідають різним подіям робочої книги.

```
'Процедура обробки події Open (відкриття книги)
Private Sub Workbook_Open()
MsgBox "Збірник прикладів VBA-програм" & Chr(13) & "Автор:
Копей В.Б."
End Sub

'Процедура обробки події BeforeClose (перед закриттям
книги)
Private Sub Workbook_BeforeClose(Cancel As Boolean)
MsgBox "VBA - найлегше програмування!"
End Sub

'Процедура обробки події SheetSelectionChange (зміна
виділення на листі)
Private Sub Workbook_SheetSelectionChange(ByVal sh As
Object, ByVal Target As Range)
Debug.Print "Зміна виділення " & Target.Address
End Sub
```

```
'Процедура обробки події SheetChange (зміна на листі)
Private Sub Workbook_SheetChange(ByVal sh As Object, ByVal
Target As Range)
Debug.Print "Зміна у " & Target.Address
End Sub

'Процедура обробки події SheetBeforeRightClick (натиск
правої кнопки миші на листі)
Private Sub Workbook_SheetBeforeRightClick(ByVal sh As
Object, ByVal Target As Range, Cancel As Boolean)
Debug.Print "Правий натиск на " & Target.Address
End Sub
```

Клас Chart - діаграма Excel

Клас Chart описує поняття діаграми Excel [19]. Об’єкт активної діаграми можна отримати за допомогою властивості ActiveChart. Наступний приклад додає дані на лист Excel і за цими даними будує діаграму з логарифмічною регресією (рис. 3.1).

```
Public Sub main()
'Додати дані на лист:
Range("A1:A6")=Application.Transpose(Array(1,2,3,4,5,6))
Range("B1:B6")=Application.Transpose(Array(0,3,6,7,9,11))
Charts.Add 'додати діаграму
ActiveChart.ChartType = xlXYScatterSmooth 'тип діаграми
ActiveChart.SetSourceData
Source:=Sheets("Лист1").Range("A1:B6"), PlotBy:= _
    xlColumns 'дані для діаграми
ActiveChart.Location Where:=xlLocationAsObject,
Name:="Лист1" 'розміщення
With ActiveChart 'з активною діаграмою
    .HasTitle = True 'має заголовок
    .ChartTitle.Characters.Text = "Графік" 'надпис
заголовка
    .Axes(xlCategory, xlPrimary).HasTitle = True 'має
надпис осі категорій
```

```
    .Axes(xlCategory, xlPrimary).AxisTitle.Characters.Text
= "x" 'надпис
    .Axes(xlValue, xlPrimary).HasTitle = True 'має надпис
осі значень
    .Axes(xlValue, xlPrimary).AxisTitle.Characters.Text =
"y" 'надпис
    'додати криву регресії, її рівняння, та величину
достовірності апроксимації
    .SeriesCollection(1).Trendlines.Add
Type:=xlLogarithmic, _
    DisplayEquation:=True, DisplayRSquared:=True
End With
End Sub
```

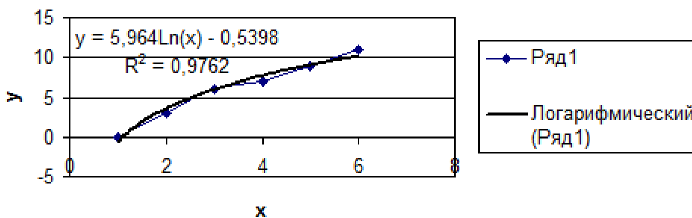


Рисунок 3.1 - Діаграма Excel

Об'єктна модель Word

Об'єктна модель Word (Microsoft Word 11.0 Object Library) дозволяє звертатись з VBA-програми до усіх основних компонентів програми Word: документів, вікон, діапазонів, абзаців тощо [15, 16, 19, 21, 29, 31-33, 45]. В прикладі показано використання основних компонентів Word, їх властивостей і методів. Перед виконанням виберіть в меню: Tools/References/Microsoft Word 11.0 Object Library.

```
Public Sub main()
Dim wdApp As Word.Application 'об'єкт програма Word
```

```
Dim wdDoc As Word.Document 'об'єкт документ
Set wdApp = CreateObject("Word.Application") 'створити
об'єкт Word
Set wdDoc = wdApp.Documents.Add 'створити об'єкт документ
wdApp.Visible = True 'зробити видимим Word
wdDoc.ActiveWindow.View.Zoom.PageFit = wdPageFitBestFit
'масштаб виду
wdDoc.ActiveWindow.Selection.TypeText "hello world!"
'надрукувати текст
wdDoc.ActiveWindow.Selection.TypeParagraph 'надрукувати
знак абзацу
wdDoc.ActiveWindow.Selection.TypeText "Visual Basic for
Applications"
'колір діапазону з перших шести символів
wdDoc.Range(0, 6).Font.Color = wdColorBlue
'виділити діапазон з перших двох слів
wdDoc.Range(wdDoc.Words(1).Start,
wdDoc.Words(2).End).Select
'розширити виділення до символу "!"
wdDoc.ActiveWindow.Selection.Extend Character:="!"
'вивести текст виділення
Debug.Print
wdApp.ActiveDocument.ActiveWindow.Selection.Text
wdDoc.ActiveWindow.Selection.Copy 'копіювати виділення в
буфер обміну
wdDoc.ActiveWindow.Selection.InsertAfter "!!" 'вставити
після виділення "!!"
'відмінити виділення і перемістити курсор в його кінець
wdDoc.ActiveWindow.Selection.Collapse
Direction:=wdCollapseEnd
wdApp.Documents.Add 'додати документ
n = wdApp.ActiveDocument.name 'ім'я активного документа
wdApp.ActiveWindow.Selection.Paste 'вставити з буфера
обміну
wdApp.Documents(2).Windows(1).Activate 'активувати другий
документ
'надати жирний шрифт першому слову першого абзацу
```

```
wdApp.Documents(2).Paragraphs(1).Range.Words(1).Font.Bold
= True
'вивести перший символ першого слова першого речення
Debug.Print
wdApp.Documents(2).Sentences(1).Words(1).Characters(1)
wdApp.Documents(n).SaveAs ("e:\my_doc.doc") 'зберегти як
wdApp.Documents("my_doc").Close 'закрити документ з іменем
my_doc
wdApp.Documents.Open ("e:\my_doc.doc") 'відкрити документ
'знайти слова "world" і замінити на "World"
With ActiveDocument.Content.Find
     .ClearFormatting
     .Text = "world"
     .Replacement.ClearFormatting
     .Replacement.Text = "World"
     .Execute Replace:=wdReplaceAll
End With
wdApp.Quit 'вийти з Word
End Sub
```

```
hello world!
h
```

OLE Automation - використання об’єктів Excel з VBA-сценарію Word

OLE Automation - це основана на COM технологія міжпроцесової взаємодії, яка дозволяє одній програмі управляти об’єктами, що реалізовані в іншій програмі [5, 20, 29]. Сервер автоматизації - це програма (COM сервер), функціональність якої доступна через COM інтерфейси (такі як IDispatch) іншим програмам, які називаються клієнтами автоматизації. Для прикладу VBA-програма (клієнт автоматизації) може використовувати елементи керування ActiveX (сервери автоматизації). Або можна використовувати об’єкти Excel з VBA-сценарію Word. Для цього виберіть в меню Tools/References/Microsoft Excel 11.0 Object Library. В програмі оголосіть Automation-об’єкт класу

`Excel.Application` і створіть його функцією `CreateObject`, якій потрібно передати програмний ідентифікатор `"Excel.Application"`. Наступний приклад створює об'єкт програми Excel, записує значення в комірку першого листа активної книги, зберігає файл і виходить з Excel.

```
Public Sub main()
Dim xlApp As Excel.Application
Dim xlBook As Excel.Workbook
Dim xlSheet As Excel.Worksheet
Set xlApp = CreateObject("Excel.Application") ' Excel
xlApp.Workbooks.Add ' нова робоча книга
Set xlBook = xlApp.ActiveWorkbook ' активна книга
'Set xlBook = xlApp.Workbooks.Open("e:\mytestbook.xls") '
або відкрити
Set xlSheet = xlBook.Worksheets(1) ' перший лист
xlSheet.Cells(1, 1).Value = 1 ' записати в комірку
'xlApp.Visible = True ' зробити видимим
xlBook.SaveAs ("e:\mytestbook.xls") 'зберегти як
'xlBook.Save ' або зберегти
xlApp.Quit ' вийти з Excel
End Sub
```

Microsoft Shell Controls And Automation

Windows shell - це графічний інтерфейс користувача операційної системи Windows. Для доступу до цього інтерфейсу з VBA програми можна використовувати об'єкти бібліотеки Microsoft Shell Controls And Automation. За допомогою цих об'єктів ви можете отримати доступ до файлової системи, виконувати програми, змінювати системні налаштування тощо. Для доступу до бібліотеки виберіть в меню: Tools/References/Browse... C:\WINDOWS\system32\SHELL32.dll.

```
Dim shl As Shell32.Shell 'об'єкт Shell
```

```
Public Sub main()
Set shl = CreateObject("Shell.Application") 'створити
об'єкт Shell
shl.ControlPanelItem "appwiz.cpl" 'відкрити "Установка і
видалення програм"
shl.Explore "c:\" 'відкрити Explorer
shl.Open "c:\windows" 'відкрити папку
shl.FileRun 'відкрити "Запуск програми"
Set Folder = shl.Namespace("d:\") 'створити об'єкт папка
d:\
Folder.CopyHere "c:\boot.ini" 'копіювати в d:\ файл
Set File = Folder.parsename("boot.ini") 'створити об'єкт
файл boot.ini
File.InvokeVerb "Open" 'відкрити файл
End Sub
```

Об'єктна модель Windows Script Host

Сервер сценаріїв Windows (Windows Script Host) — компонент Windows, призначений для запуску програм мовами сценаріїв (JScript, VBScript). Об'єктна модель Windows Script Host дозволяє звертатись до сервера сценаріїв з метою виконання адміністративних задач [19, 20]. Виберіть в меню Tools/References/Browse... C:\WINDOWS\system32\wshom.ocx і Windows Script Host Object Model.

```
Dim sh As IWshRuntimeLibrary.WshShell 'об'єкт WshShell

Public Sub main()
Set sh = CreateObject("WScript.Shell") 'створення об'єкта
WshShell
sh.Run "cmd.exe /c net share d=d:\" 'виконати cmd.exe з
параметрами
sh.Run "cmd.exe /k net share d /delete"
app = Shell("calc.exe", 1) 'виконати calc.exe
sh.AppActivate "Calculator" 'активувати вікно (або так:
AppActivate app)
```

```
sh.SendKeys "1{+}2{Enter}", True 'надіслати вікну клавіші
'прочитати з реєстру
Debug.Print sh.RegRead("HKEY_CURRENT_USER\Control
Panel\Desktop\ScreenSaveTimeOut")
'записати в реєстр
sh.RegWrite "HKEY_CURRENT_USER\Control
Panel\Desktop\ScreenSaveTimeOut", _
"600", "REG_SZ"
Debug.Print sh.Popup("text", 0, "title", 1) 'показати
діалогове вікно
Debug.Print sh.ExpandEnvironmentStrings("%WinDir%") 'шлях
до папки Windows
sh.LogEvent 1, "Hello!" 'записує повідомлення з помилкою в
системний журнал
End Sub
```

Об'єкти файлової системи в Windows Script Host Object Model

Наступний приклад показує використання об’єктів файлової системи (диски, файли, текстові потоки) за допомогою об’єктної моделі Windows Script Host.

```
Dim fso As IWshRuntimeLibrary.FileSystemObject 'об'єкт
файлової системи
Dim drvs As IWshRuntimeLibrary.Drives 'диски
Dim drv As IWshRuntimeLibrary.Drive 'диск
Dim fl As IWshRuntimeLibrary.File 'файл
Dim ts As IWshRuntimeLibrary.TextStream 'текстовий потік

Public Sub main()
'об'єкт файлової системи
Set fso = CreateObject("Scripting.FileSystemObject")
Set drvs = fso.Drives 'диски
Debug.Print drvs.Count 'вивести кількість дисків
For Each drv In drvs 'для кожного диску
 'якщо тип диску Removable і він доступний, тоді
 If drv.DriveType = Removable And drv.IsReady Then
```

```
    Debug.Print drv.DriveLetter 'вивести букву диска
    'для кожного файлу в кореневій директорії
    For Each fl In drv.RootFolder.Files
        If fl.name = "Autorun.inf" Then 'якщо ім'я
"Autorun.inf", то
            fl.Copy "d:\" 'копіювати на диск d
            'відкрити файл для читання
            Set ts = fl.OpenAsTextStream(ForReading)
            Debug.Print ts.ReadAll 'читати все
            ts.Close 'закрити
            fl.name = "Autorun_.inf" 'змінити ім'я
        End If
    Next
 End If
Next
End Sub
```

Виконання сценаріїв Windows Script Host

В прикладі показано створення і виконання сценаріїв мовами Jscript і VBscript з VBA-програми. Файл сценарію зберігається на диску і виконується за допомогою сервера сценаріїв (wscript.exe або cscript.exe). Сервер сценаріїв запускається з VBA-програми за допомогою методу `Run` об’єкта `IWshRuntimeLibrary.WshShell`.

```
Dim sh As IWshRuntimeLibrary.WshShell 'об'єкт WshShell

Public Sub main()
Set sh = CreateObject("WScript.Shell") 'створити об'єкт
WshShell
'
_____________________________________________________
'Створення файлу сценарію мовою Jscript
Open "c:\my.js" For Output As #1 'відкрити файл для запису
Print #1, "var objArgs = WScript.Arguments;" 'записати у
файл
Print #1, "for (i=0; i<=objArgs.Count()-1;i++)" 'записати
у файл
```

```
Print #1, "{WScript.Echo(parseFloat(objArgs(i))+2);}"
'записати у файл
Close #1 'закрити файл
sh.Run "wscript.exe c:\my.js 1 2" 'виконання сценарію з
параметрами
'
_____________________________________________________
'Створення файлу сценарію мовою VBscript
Open "c:\my.vbs" For Output As #1 'відкрити файл для
запису
Print #1, "For Each obj In WScript.Arguments" 'записати у
файл
Print #1, "x=CDbl(WScript.StdIn.Readline)" 'записати у
файл
Print #1, "WScript.StdOut.WriteLine obj+x" 'записати у
файл
Print #1, "Next" 'записати у файл
Close #1 'закрити файл
sh.Run "cscript.exe c:\my.vbs 1 2" 'виконання в консолі
сценарію з параметрами
'
_____________________________________________________
'Створення файлу сценарію XML мовами VBscript і Jscript
Open "c:\my.wsf" For Output As #1 'відкрити файл для
запису
Print #1, "<package>"
Print #1, "<job id=""myjob"">" 'один пакет може містити
кілька робіт
Print #1, "<script language=""VBScript"">"
Print #1, "Function F"
Print #1, "WScript.Echo WScript.ScriptFullName"
Print #1, "End Function"
Print #1, "</script>"
Print #1, "<script language=""JScript"">"
Print #1, "F();" 'виклик функції, написаній на VBscript
Print #1, "</script>"
Print #1, "</job>"
Print #1, "</package>"
Close #1 'закрити файл
```

```
sh.Run "wscript.exe //job:""myjob"" c:\my.wsf" 'виконання
сценарію
End Sub
```

Параметричні моделі у SOLIDWORKS API

SOLIDWORKS - це відома комерційна система автоматизованого проектування (САПР), яка призначена для тривимірного параметричного проектування виробів, підготовки креслень та інженерного аналізу. Об'єктна модель і API SOLIDWORKS дозволяє звертатись до компонентів програми з мови VBA [8, 9, 12, 14, 27]. Багато інших САПР теж мають API для VBA, зокрема CATIA, Femap, Inventor [2], AutoCAD [24]. Приклад показує можливість перебудови параметричної моделі SOLIDWORKS, яка містить параметр `"D1@Extrude1"`. Щоб створити цей сценарій в SOLIDWORKS виберіть в меню Tools/Macro/New... Для доступу до програми SOLIDWORKS в сценарії використовується об'єкт `swApp`. Для доступу до моделі використовується об'єкт `Part`. Значення параметра (в метрах) присвоюються властивості `SystemValue` параметра. Після зміни значення параметра модель потрібно перебудувати функцією `EditRebuild`. Цей приклад можна адаптувати для автоматизації зміни будь-яких параметричних моделей, в тому числі вузлів.

```
Public Sub main()
Dim swApp As Object 'об'єкт SolidWorks Application
Dim Part As Object 'об'єкт документ SolidWorks
Set swApp = CreateObject("SldWorks.Application") 'створити
об'єкт
Set Part = swApp.ActiveDoc 'активний документ
'змінити значення параметра на 20 мм
Part.Parameter("D1@Extrude1").SystemValue = 20 / 1000
Part.EditRebuild 'перебудувати модель
End Sub
```

Параметричні моделі з рівняннями у SOLIDWORKS API

Приклад показує можливість перебудови параметричної моделі SOLIDWORKS, яка містить рівняння (рис. 3.2), шляхом зміни цих рівнянь. Модель містить чотири рівняння (рис. 3.3), одне з яких є глобальною змінною `h`. Рівняння в розділі Features дозволяє вимкнути елемент Fillet1 (скруглення). Решта рівнянь задають розміри моделі - довжину (`"D2@Sketch1"`) і товщину (`"D1@Boss-Extrude1"`). Зверніть увагу, що параметр висоти моделі не залежить від рівнянь.

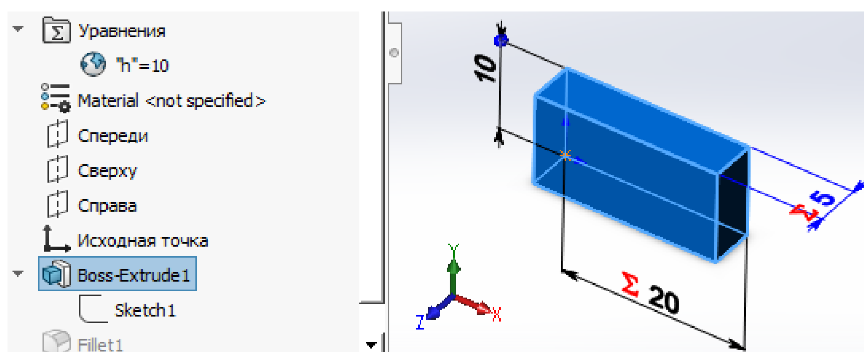


Рисунок 3.2 - Параметрична модель з рівняннями

Name	Value / Equation
Global Variables	
"h"	= 10.0
Add global variable	
Features	
"Fillet1"	= "suppressed"
Add feature suppression	
Equations	
"D2@Sketch1"	= "h" * 2
"D1@Boss-Extrude1"	= "h" / 2
Add equation	

Рисунок 3.3 - Рівняння моделі

```
Option Explicit
```

```
Sub main()
    Dim swApp As SldWorks.SldWorks
    Dim swModel As SldWorks.ModelDoc2
    Dim swEqnMgr As SldWorks.EquationMgr 'менеджер рівнянь
    Dim i As Long 'індекс рівняння
    Dim nCount As Long 'кількість рівнянь

    Set swApp = Application.SldWorks
    Set swModel = swApp.ActiveDoc 'активний документ
    Set swEqnMgr = swModel.GetEquationMgr 'менеджер
рівнянь
    Debug.Print "File = " & swModel.GetPathName 'шлях до
моделі
    nCount = swEqnMgr.GetCount 'кількість рівнянь
    swEqnMgr.Equation(0) = """h"" = 10.0" 'змінити перше
    For i = 0 To nCount - 1 'для кожного рівняння вивести
інформацію
        Debug.Print "  Equation(" & i & ")  = " &
swEqnMgr.Equation(i)
        Debug.Print "    Value = " & swEqnMgr.Value(i)
        Debug.Print "    Index = " & swEqnMgr.Status
        Debug.Print "    Global variable? " &
swEqnMgr.GlobalVariable(i)
    Next i
End Sub
```

```
File = D:\EquationAPItest.SLDPRT
  Equation(0)  = "h" = 10.0
    Value = 10
    Index = 0
    Global variable? True
  Equation(1)  = "D2@Sketch1"= "h" * 2
    Value = 20
    Index = 1
    Global variable? False
  Equation(2)  = "D1@Boss-Extrude1"= "h" / 2
    Value = 5
    Index = 2
```

```
    Global variable? False
  Equation(3)  = "Fillet1" = "suppressed"
    Value = 1
    Index = 3
    Global variable? False
```

Симуляція кінематики у SOLIDWORKS API

Під час проектування складних механізмів часто постає питання аналізу їх кінематичних характеристик – визначення переміщень, швидкостей, прискорень точок механізму в заданий момент часу. Параметричні тривимірні моделі механізмів в SOLIDWORKS дозволяють оперативно змінювати параметри механізму, які визначають його положення в заданий момент часу. Використовуючи засоби SOLIDWORKS API можливо розробити таку програму, яка самостійно виконує: зміну необхідних параметрів механізму, визначення положення ланок і їх табулювання в таблиці Excel. Засобами Excel можна проаналізувати отримані дані (наприклад залежність переміщень точок механізмів від часу) і побудувати потрібні графіки.

Нижче наведено приклад програми мовою VBA для аналізу кінематики кривошипно-кулісного механізму поперечно-строгального верстата. Необхідно під час рівномірного руху кривошипа знайти кінематичні характеристики повзуна: переміщення, швидкість і прискорення в заданий момент часу.

Із законів механіки відомо, що швидкість і прискорення визначаються як

$$v=\frac{\Delta x}{\Delta t}; \quad a=\frac{\Delta v}{\Delta t},$$

де Δx – елементарне переміщення за час Δt, Δv – елементарна зміна швидкості за час Δt.

Спочатку будуємо тривимірну параметричну модель механізму в SOLIDWORKS (рис. 3.4). Окрім всіх необхідних зв’язків накладаємо додатковий зв’язок, який визначатиме положення

кривошипа в даний момент часу. Проставляємо також розмір *L*, який визначає положення повзуна в заданий момент часу.

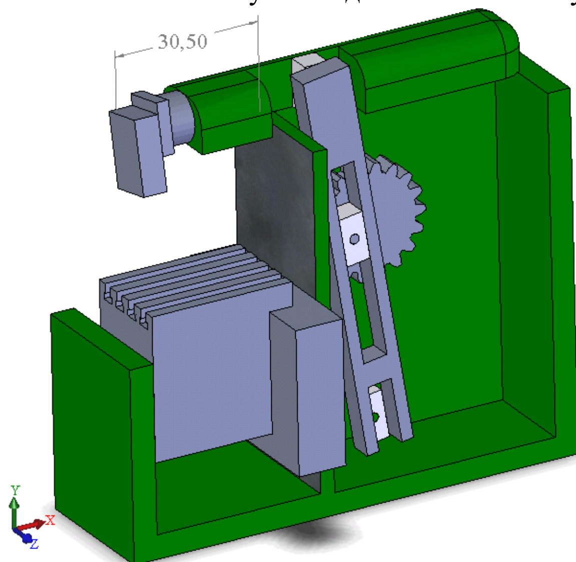


Рисунок 3.4 - Тривимірна параметрична модель кривошипно-кулісного механізму поперечно-строгального верстата

Вся робота з SW API зводиться до зміни положення кривошипа і визначення положення повзуна. Засобами Excel можна визначати швидкості і прискорення точок механізму, використовуючи згадані формули.

Діалогове вікно програми мовою VBA повинно містити текстові поля для введення повного імені параметра, який задає положення привідної ланки (кривошипа), наприклад D1@УГОЛ2, та параметра положення шуканої точки, наприклад **RD1@Примечания.** Це розширює можливості програми і робить її універсальною. В діалоговому вікні слід ввести початкове, кінцеве положення привідної ланки та крок зміни її значення. Слід також ввести спосіб задання значень вхідних параметрів - за допомогою введення значень у форму або в таблицю Excel. В останньому випадку

можливо розраховувати кінематичні параметри для заданого закону руху привідної ланки.

Рисунок 3.5 - Вікно програми

Програма складається з процедури ініціалізації ділового вікна (рис. 3.5) `UserFormInitialize()`, в якій починається робота з сесією SOLIDWORKS, завантажується збірка SOLIDWORKS та створюється об’єкт `Part`. Тут також встановлюються початкові властивості елементів ділового вікна.

Процедура `P_form()` виконує табулювання кінематичних параметрів для заданих у формі параметрів переміщень привідної ланки. Отримуються значення потрібних параметрів з текстових полів ділового вікна. В циклі змінюється переміщення привідної ланки від початкового до кінцевого значення, виконується перебудова моделі, отримується переміщення шуканої точки і записується у таблицю Excel.

Процедура `P_tab ()` виконує подібні функції, але значення переміщень отримуються не в циклі, а шляхом зчитування попередньо введених даних в комірках Excel.

Коли натискається кнопка `CommandButton1` розпочинається процедура `CommandButton1_Click()`, в якій перевіряється значення перемикачів `OptionButton`. Якщо включений `OptionButton1` то виконується процедура `P_form`, а якщо `OptionButton2` - процедура `P_tabl`.

Нехай необхідно розрахувати кінематичні характеристики, якщо кривошип рухається рівномірно з швидкістю 0,2 рад/с. В діалоговому вікні вводимо назву вхідного і вихідного параметрів, початкове і кінцеве положення кривошипа і крок зміни. Після натискання кнопки "Табулювання" в таблицю Excel будуть записані значення переміщення кривошипа і визначені з параметричної моделі відповідні значення переміщення повзуна. По цим даним Excel автоматично розрахує значення швидкостей і прискорень і побудує графік шуканої кінематики повзуна (рис. 3.6).

Дана програма може бути використана для розрахунку кінематичних параметрів і більш складних механізмів, і не тільки плоских. Слід зауважити, що в SOLIDWORKS існує спеціалізований модуль Motion для аналізу кінематики і динаміки механізмів з жорсткими ланками.

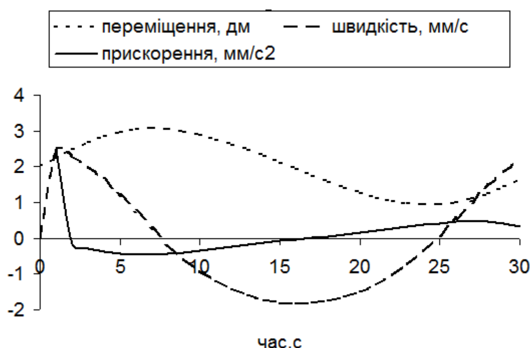


Рисунок 3.6 - Кінематика повзуна поперечно-строгального верстата

```
Const swDocPART = 1 'константи
\SldWorks\samples\appComm\swconst.h
Const swDocASSEMBLY = 2
Const swDocDRAWING = 3
Dim swApp As Object 'оголошення об'єкта swApp
Dim Part As Object 'оголошення об'єкта Part
Dim x As Double 'оголошення змінної параметра моделі
Dim i As Integer 'оголошення змінної лічильника записів
бази даних
'Ініціалізується форма користувача
Private Sub UserForm_Initialize()
    MyPath = CurDir 'визначення поточного каталогу
    MyPath = "C:\sldworks\ПСВ"
    'створення об’єкта swApp
    Set swApp = CreateObject("SldWorks.Application")
    'створення об’єкта Part
    Set Part = swApp.OpenDoc(MyPath + "\ПСВ.SLDASM",
swDocASSEMBLY)
    If Part Is Nothing Then
       Exit Sub
       'якщо Part не існує, то вихід
    Else
      'інакше активувати
       Set Part = swApp.ActivateDoc("ПСВ.SLDASM")
    End If
    'початкові параметри елементів діалогового вікна
    TextBox1.Text = "D1@Угол2"
    TextBox5.Text = "RD1@Примечания"
    OptionButton1.Value = True
    TextBox2.Text = "0"
    TextBox3.Text = "1"
    TextBox4.Text = "0,1"
End Sub
'Натиснута кнопка CommandButton1
Private Sub CommandButton1_Click()
'якщо включений перемикач OptionButton1, то p_form
```

```
 If OptionButton1.Value = True Then p_form
'якщо включений перемикач OptionButton2, то p_tabl
 If OptionButton2.Value = True Then p_tabl
End Sub
'Процедура виконує табулювання кінематичних параметрів для
заданих у формі параметрів переміщень
Private Sub p_form()
Dim p1, p2 As String
    Dim xp, xk, xd As Double
    i = 3 'початкове значення лічильника
    'отримати дані з текстових полів
    p1 = TextBox1.Text
    p2 = TextBox5.Text
    xp = CDbl(TextBox2.Text)
    xk = CDbl(TextBox3.Text)
    xd = CDbl(TextBox4.Text)
    'очистити діапазон
    Range("b3:b1000").Clear
    Range("e3:e1000").Clear
    'зміна значень параметрів
    For x = xp To xk Step xd
    Part.Parameter(p1).SystemValue = x
    Part.EditRebuild 'Перебудова моделі
    'Виведення значень параметрів у відповідні комірки
Excel
    Cells(i, 2).Value = x
    Cells(i, 5).Value = Part.Parameter(p2).SystemValue *
1000
    i = i + 1 'зміна значення лічильника
    Next x
End Sub
'Процедура виконує табулювання кінематичних параметрів для
заданих у таблиці параметрів переміщень, швидкостей і
прискорень
Private Sub p_tabl()
   Dim p1, p2 As String
    'отримати текст з текстових полів
```

```
    p1 = TextBox1.Text
    p2 = TextBox5.Text
    'очистити діапазон
    Range("e3:e1000").Clear
    'цикл від трьох до кількості чисел в діапазоні B
    For i = 3 To Cells(1, 8).Value
    'зміна значень параметрів
    Part.Parameter(p1).SystemValue = Cells(i, 2).Value
    Part.EditRebuild 'перебудова моделі
    'Виведення значень параметрів у відповідні комірки
Excel
    Cells(i, 5).Value=Part.Parameter(p2).SystemValue*1000
    Next i
End Sub
'Процедура закриття діалогового вікна
Private Sub CommandButton2_Click()
UserForm1.Hide
End Sub
```

Креслення у SOLIDWORKS API

Наступна програма являє собою макрос SOLIDWORKS 2015 мовою VBA та призначена для автоматизації побудови нового та перебудови існуючого креслення SOLIDWORKS за даними структури типу `DrawingData`. Програма складається з модулів `DrawingView`, `DrawingSheet`, `DrawingData`, `Macro21`, `UserForm1` (рис. 3.7). Модуль `DrawingView` містить опис класу виду, модуль `DrawingSheet` – класу листа, модуль `DrawingData` – класу, що описує дані усього креслення. Модуль `Macro21` містить реалізацію процедур для побудови креслення за даними `DrawingData` (`GenerateDrawing`), перебудови креслення (`UpdateDrawing`), головну процедуру (`main`) та інші допоміжні функції. Робота програми починається з процедури `main`, де створюється тестовий об’єкт класу `DrawingData` та викликаються процедури `GenerateDrawing` або `UpdateDrawing`. Іншим способом роботи з програмою є виконання модуля `UserForm1`,

який створює вікно для введення та редагування даних DrawingData (рис. 3.7). У вікні можна вибрати потрібний лист і вид, змінити значення атрибутів виду, зберегти їх (кнопка Save), створити креслення (конопка GenerateDrawing) або перебудувати існуюче (кнопка UpdateDrawing).

Для роботи програмі необхідна SOLIDWORKS 2015-2018. В інших версіях програма не тестувалась. Послідовність установки і виконання:

1. Скопіювати в довільне місці на диску каталог, який містить наступні файли:

- updateDrawing.swp – макрос SOLIDWORKS мовою VBA,
- Drawing1.SLDDRW – креслення, яке містить згенеровані і додані вручну види (рис. 3.9),
- Part1.SLDPRT – довільна деталь з двома конфігураціями (рис. 3.8),
- Part2.SLDPRT – довільна деталь з двома конфігураціями,
- a3-gost_sh1.slddrt – шаблон листа креслення.

2. Виконати макрос з SOLIDWORKS так: Tools\Macro\Run... та вибрати макрос updateDrawing.swp. Зявиться вікно де потрібно вибрати «Да» для генерації креслення та «Нет» для обновлення креслення Drawing1.SLDDRW (рис. 3.9).

3. Іншим способом виконання програми (з GUI) є вибір в SOLIDWORKS меню Tools\Macro\Edit... та вибір макроса updateDrawing.swp. Далі потрібно двічі клацнути на модулі UserForm1 та виконати його (меню Run\Run UserForm).

Макрос розробив студент групи КМВт-15-1 Букатка Ю. С. під керівництвом доцента Копея В. Б. під час виконання завдання "SolidWorksAPITask" на 18-й заочній Міжнародній студентській олімпіаді з САПР і комп’ютерного моделювання в машинобудуванні (1-12 грудня 2016 р., Хмельницький національний університет).

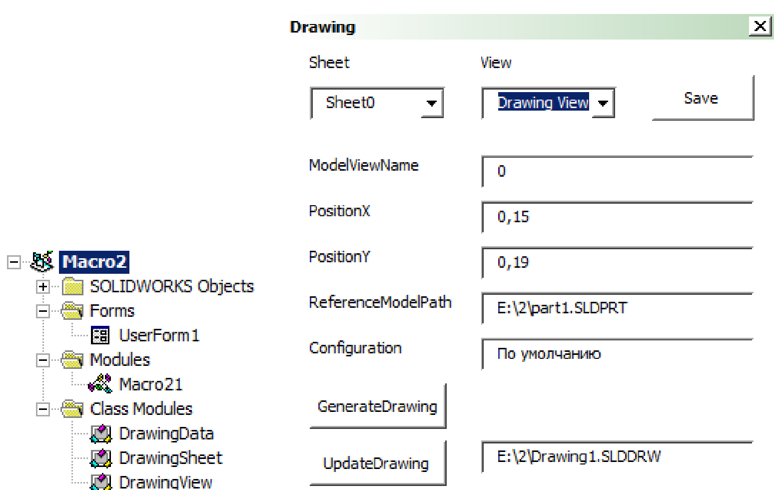


Рисунок 3.7 – Структура макроса і вікно програми

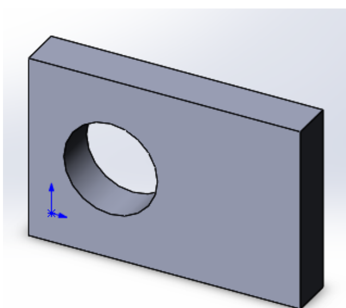


Рисунок 3.8 - Модель Part1

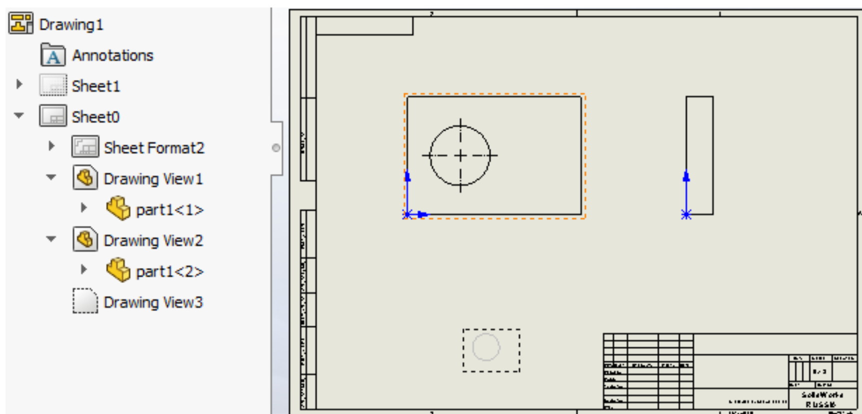


Рисунок 3.9 - Результати виконання програми - креслення з трьома видами

Модуль класу DrawingData:

```
'дані класу
Public Sheets As Collection
Public Name As String
Private Sub Class_Initialize() 'процедура ініціалізації
Name = "drawingData"
Set Sheets = New Collection
End Sub
```

Модуль класу DrawingSheet:

```
Public views As Collection
Public Name As String
Private Sub Class_Initialize()
Name = ""
Set views = New Collection
End Sub
```

Модуль класу DrawingView:

```
Public Name As String
Public ModelViewName As Integer '0-*Front,1-*Front,2-
*Back,3-*Left,4-*Right,5-*Top,6-*Bottom
Public PositionX As Double
```

```
Public PositionY As Double
Public ReferenceModelPath As String
Public Configuration As String

Private Sub Class_Initialize()
Name = ""
ModelViewName = 0
PositionX = 0
PositionY = 0
ReferenceModelPath = ""
Configuration = ""
End Sub
```

Модуль Macro21:

```
Dim swApp As SldWorks.SldWorks 'об'єкт програма-SolidWorks
Dim swModel As SldWorks.ModelDoc2 'об'єкт модель
Dim swDraw As SldWorks.DrawingDoc 'об'єкт креслення
Dim myView As SldWorks.view 'об'єкт вид
Dim swModelDocExt As SldWorks.ModelDocExtension 'об'єкт
розширення моделі
Dim swDrawingComponent As SldWorks.DrawingComponent
'об'єкт компонент креслення
Dim swSelectionMgr As SldWorks.SelectionMgr 'об'єкт
менеджера вибору

Public macroPath As String 'шлях до каталогу з макросом
(робочого каталогу)
Dim boolstatus As Boolean
Dim longstatus As Long, longwarnings As Long

'процедура створює креслення за даними data
Sub GenerateDrawing(data As DrawingData)
'масив імен стандартних видів
ViewNames = Array("*Front", "*Front", "*Back", "*Left",
"*Right", "*Top", "*Bottom")

Set swApp = Application.SldWorks
```

```
'нове креслення
Set swDraw = swApp.NewDocument(macroPath & "\\a3 -
gost_sh1.slddrt", 0, 0, 0)
For Each s In data.Sheets 'для кожного листа в data
    'новий лист
    boolstatus = swDraw.NewSheet3(s.Name, 12, 12, 2, 1,
True, macroPath & "\\a3 - gost_sh1.slddrt", 0.42, 0.297,
"По умолчанию")
    For Each v In s.views 'для кожного виду листа в data
        'новий вид
        Set myView =
swDraw.CreateDrawViewFromModelView3(v.ReferenceModelPath,
ViewNames(v.ModelViewName), v.PositionX, v.PositionY, 0)
        'задати конфігурацію
        myView.ReferencedConfiguration = v.Configuration
    Next v
Next s

End Sub

'оновлює креслення drawingPathName за даними data
Public Sub UpdateDrawing(drawingPathName As String, data
As DrawingData)
Dim sname As String
Dim compName As String

Set swApp = Application.SldWorks
'відкрити креслення
Set swDraw = swApp.OpenDoc6(drawingPathName, 3, 0, "",
longstatus, longwarnings)

snames = swDraw.GetSheetNames 'назви листів креслення
For i = 0 To swDraw.GetSheetCount - 1 'для кожного листа
    sname = snames(i) 'назва листа
    swDraw.ActivateSheet sname 'активувати лист
    Debug.Print sname
```

```
    vs = swDraw.Sheet(sname).GetViews 'види листа
    If TypeName(vs) <> "Empty" Then 'якщо є види
        For Each v In vs 'для кожного виду
            swDraw.ActivateView v.Name 'активувати вид
            Debug.Print v.Name, v.GetReferencedModelName
'назва виду і його вихідної моделі
            compName = ""
            If TypeName(v.GetVisibleComponents) <> "Empty"
Then
                For Each c In v.GetVisibleComponents 'для
кожного видимого компонента
                    compName = c.Name 'назва компонента
                Next c
            End If
            pos = v.Position 'позиція виду
            Set myView = v 'об'єкт виду (необхідно для
безпомилкової роботи функції GetOrientationName)
            Debug.Print myView.GetOrientationName 'назва
орієнтації виду (*Front, *Left...)

            Set dv = findView(sname, v.Name, data) 'знайти
вид у data
            If Not dv Is Nothing Then 'якщо є
                Debug.Print v.GetReferencedModelName,
dv.ReferenceModelPath
                'якщо назви вихідної моделі не однакові
                If v.GetReferencedModelName <>
dv.ReferenceModelPath Then
                    Debug.Print "ReplacingViewModel..."
                    'замінити вихідну модель на нову
                    ReplaceViewModel sname, v.Name,
compName, dv.ReferenceModelPath
                End If
                pos(0) = dv.PositionX
                pos(1) = dv.PositionY
                v.Position = pos 'нова позиція
                v.ReferencedConfiguration =
```

```
dv.Configuration 'нова конфігурація

                'вибрати вид
                boolstatus =
swDraw.Extension.SelectByID2(v.Name, "DRAWINGVIEW", 0, 0,
0, False, 0, Nothing, 0)
                swDraw.ShowNamedView2 "", dv.ModelViewName
'нова орієнтація виду, або swDraw.ShowNamedView2 "*Right",
-1
            End If
        Next v
    End If
Next i

swDraw.EditRebuild 'перебудувати модель
End Sub

'функція шукає вид у data
Public Function findView(sheetName As String, viewName As
String, data As DrawingData) As DrawingView
Set findView = Nothing
For Each s In data.Sheets 'для кожного листа
    If s.Name = sheetName Then
        For Each v In s.views 'для кожного виду
            If v.Name = viewName Then
                Set findView = v 'функція повертає вид або
Nothing
            End If
        Next v
    End If
Next s
End Function

'процедура заміни вихідної моделі у виді креслення на нову
modelFileName
Public Sub ReplaceViewModel(sheetName As String, viewName
As String, componentName As String, modelFileName As
```

```
String)
Dim views(0) As Object 'вид
Dim instances(0) As Object 'компонент
Set swModel = swApp.ActiveDoc 'креслення
Set swModelDocExt = swModel.Extension
vs = swDraw.Sheet(sheetName).GetViews 'усі види
For Each v In vs 'для кожного виду
    If v.Name = viewName Then 'знайти вид з іменем
viewName
        Set views(0) = v
    End If
Next v
id_ = componentName & "@" & viewName
'вибрати компонент
boolstatus = swModelDocExt.SelectByID2(id_, "COMPONENT",
0, 0, 0, False, 0, Nothing, 0)
Set swSelectionMgr = swModel.SelectionManager
Set swDrawingComponent =
swSelectionMgr.GetSelectedObject6(1, -1)
Set instances(0) = swDrawingComponent.Component
'замінити модель
boolstatus = swDraw.ReplaceViewModel(modelFileName,
(views), (instances))
End Sub

Public Sub main() 'процедура для тестування
GenerateDrawing та UpdateDrawing
'створити об'єкти
Dim view1 As New DrawingView
Dim view2 As New DrawingView
Dim sheet1 As New DrawingSheet
Dim sheet2 As New DrawingSheet
Dim data As New DrawingData

Set swApp = Application.SldWorks
macroPath = UCase(swApp.GetCurrentMacroPathFolder) 'шлях
до каталогу з макросом
```

```
Debug.Print macroPath

'присвоїти атрибутам значення
view1.Name = "Drawing View1"
view2.Name = "Drawing View2"
view1.ReferenceModelPath = macroPath & "\part1.SLDPRT"
view2.ReferenceModelPath = macroPath & "\part1.SLDPRT"
view1.Configuration = "По умолчанию"
view2.Configuration = "По умолчанию"
view1.ModelViewName = 0
view2.ModelViewName = 3
sheet1.Name = "Sheet0"
sheet2.Name = "Sheet1"
view1.PositionX = 0.15
view1.PositionY = 0.19
view2.PositionX = 0.3
view2.PositionY = 0.19

sheet1.views.Add view1 'додати види в листи
sheet1.views.Add view2
data.Sheets.Add sheet1 'додати листи в data
data.Sheets.Add sheet2

'Set dv = findView("Sheet0", "Drawing View1", data) ' тест
функції пошуку
'If Not dv Is Nothing Then Debug.Print
dv.ReferenceModelPath

'вікно з кнопками Yes-No
nvar = MsgBox("YES for Generate Drawing" & Chr(13) & "or
NO for Update Drawing", 4, "Press")
Debug.Print nvar
If nvar = 6 Then 'якщо вибрано Yes
    GenerateDrawing data 'створити креслення
Else
    'інакше змінити атрибути і оновити креслення
    view1.ReferenceModelPath = macroPath & "\part2.SLDPRT"
```

```
    view2.ReferenceModelPath = macroPath & "\part2.SLDPRT"
    view2.Configuration = "20"
    view1.ModelViewName = 2
    view1.PositionX = 0.14
    view1.PositionY = 0.18
    UpdateDrawing macroPath & "\Drawing1.SLDDRW", data
'оновити
End If
End Sub
```

Модуль UserForm1:

```
Dim swApp As SldWorks.SldWorks
Dim view As New DrawingView
Dim data As New DrawingData

Private Sub ComboBox1_Change() 'вибрано лист
ComboBox2.Clear
For Each s In data.Sheets
    If s.Name = ComboBox1.Text Then
        For Each v In s.views
            ComboBox2.AddItem v.Name
        Next v
    End If
Next s
End Sub

Private Sub ComboBox2_Change() 'вибрано вигляд
Set view = findView(ComboBox1.Text, ComboBox2.Text, data)
writeViewFields
End Sub

Private Sub CommandButton2_Click() 'кнопка save
Set view = findView(ComboBox1.Text, ComboBox2.Text, data)
readViewFields
End Sub

Private Sub CommandButton1_Click() 'кнопка GenerateDrawing
```

```
GenerateDrawing data
End Sub

Private Sub CommandButton3_Click() 'кнопка UpdateDrawing
UpdateDrawing TextBox6.Text, data
End Sub

Private Sub UserForm_Initialize() 'ініціалізація форми
Dim view1 As New DrawingView
Dim view2 As New DrawingView
Dim sheet1 As New DrawingSheet
Dim sheet2 As New DrawingSheet

Set swApp = Application.SldWorks
macroPath = UCase(swApp.GetCurrentMacroPathFolder)
Macro21.macroPath = macroPath 'шлях до каталогу з макросом
Debug.Print macroPath

'заповнити дані
view1.Name = "Drawing View1"
view2.Name = "Drawing View2"
view1.ReferenceModelPath = macroPath & "\part1.SLDPRT"
view2.ReferenceModelPath = macroPath & "\part1.SLDPRT"
view1.Configuration = "По умолчанию"
view2.Configuration = "По умолчанию"
view1.ModelViewName = 0
view2.ModelViewName = 3
sheet1.Name = "Sheet0"
sheet2.Name = "Sheet1"
view1.PositionX = 0.15
view1.PositionY = 0.19
view2.PositionX = 0.3
view2.PositionY = 0.19

sheet1.views.Add view1
sheet1.views.Add view2
data.Sheets.Add sheet1
```

```
data.Sheets.Add sheet2

'заповнити список листів
For Each s In data.Sheets
    ComboBox1.AddItem s.Name
Next s

TextBox6.Text = macroPath & "\Drawing1.SLDDRW"

End Sub

Public Sub writeViewFields() 'записати дані поточного виду
в поля
ComboBox2.Text = view.Name
TextBox1.Text = view.ModelViewName
TextBox2.Text = view.PositionX
TextBox3.Text = view.PositionY
TextBox4.Text = view.ReferenceModelPath
TextBox5.Text = view.Configuration
End Sub

Public Sub readViewFields() 'прочитати дані поточного виду
з полів
view.Name = ComboBox2.Text
view.ModelViewName = CInt(TextBox1.Text)
view.PositionX = CDbl(TextBox2.Text)
view.PositionY = CDbl(TextBox3.Text)
view.ReferenceModelPath = TextBox4.Text
view.Configuration = TextBox5.Text
End Sub
```

SOLIDWORKS Simulation API

Нижче наведено макрос для SOLIDWORKS Simulation 2018, який розраховує коефіцієнт запасу втомної міцності за критерієм Сайнса. Для роботи макросу потрібні результати двох статичних задач (кроків) в SOLIDWORKS Simulation - для мінімального

(перший крок) і максимального (другий крок) навантаження циклу. Функція FOS розраховує коефіцієнт запасу втомної міцності за відомими значеннями головних напружень на першому кроці (S1_1, S2_1, S3_1) і другому кроці (S1_2, S2_2, S3_2). Значення границі витривалості зберігається в локальній змінній sn (МПа). Потрібно також ввести номери вузлів, в яких будуть розраховуватись значення FOS. В даному прикладі це вузли 6313, 6334, 6349, 198, 186. Перед запуском необхідно вибрати в меню Tools/References... бібліотеку SOLIDWORKS Simulation 2018 type library. Макрос можна легко адаптувати і для інших задач. Наприклад для автоматизації перебудови сітки і запуску обчислення використовуйте методи Study.MeshAndRun, Study.CreateMesh(), Study.RunAnalysis.

```
Attribute VB_Name = "Macro11"
Dim swApp As Object
Dim COSMOSWORKS As Object
Dim CWObject As CosmosWorksLib.CwAddincallback
Dim ActDoc As CosmosWorksLib.CWModelDoc
Dim StudyMngr As CosmosWorksLib.CWStudyManager
Dim Study As CosmosWorksLib.CWStudy
Dim errCode As Long 'код помилки
Dim CWResult As CosmosWorksLib.cwResults
'Dim LBCMgr As CosmosWorksLib.CWLoadsAndRestraintsManager
'Dim lr As CosmosWorksLib.CWLoadsAndRestraints
Dim Face As SldWorks.Face2

'Розраховує коефіцієнт запасу втомної міцності за
критерієм Сайнса. S1_2 - головне напруження 1 крок 2
(максимальне навантаження), 'S1_1 - головне напруження 1
крок 1 (мінімальне навантаження), МПа
Public Function FOS(S1_2, S1_1, S2_2, S2_1, S3_2, S3_1 As
Double)
    sn = 207 '000000 'границя витривалості
    m = 1 'коефіцієнт
```

```
    Sm3 = (S3_2 + S3_1) / 2
    Sa3 = (S3_2 - S3_1) / 2

    Sm2 = (S2_2 + S2_1) / 2
    Sa2 = (S2_2 - S2_1) / 2

    Sm1 = (S1_2 + S1_1) / 2
    Sa1 = (S1_2 - S1_1) / 2

    FOS = (sn - m * (Sm1 + Sm2 + Sm3) / 3) / Sqr(((Sa1 -
Sa2) ^ 2 + (Sa2 - Sa3) ^ 2 + (Sa3 - Sa1) ^ 2) / 2)
End Function

Sub main()
Dim s1(1 To 2), s2(1 To 2), s3(1 To 2) As Double 'головні
напруження
Set swApp = Application.SldWorks 'об'єкт Solidworks
Set CWObject = swApp.GetAddInObject("SldWorks.Simulation")
'об'єкт Simulation
Set COSMOSWORKS = CWObject.COSMOSWORKS
Set ActDoc = COSMOSWORKS.ActiveDoc() 'активний документ
COSMOSWORKS
Set StudyMngr = ActDoc.StudyManager() 'менеджер задач
Set Part = swApp.ActiveDoc 'активна деталь

For Each N In Array(6313, 6334, 6349, 198, 186) 'вузол
For i = 1 To 2
StudyMngr.ActiveStudy = i - 1
Set Study = StudyMngr.GetStudy(i - 1) 'задача

'Study.MeshAndRun
'errCode = Study.CreateMesh(0, 4.7, 0.25) 'створити сітку
'runError = Study.RunAnalysis 'виконати задачу
Set CWResult = Study.results 'результати

MaxStep = CWResult.GetMaximumAvailableSteps()
sn = CWResult.GetStress(0, MaxStep, Nothing, 3, errCode)
```

```
'масив напружень у вузлах

s1(i) = sn((N - 1) * 12 + 7) 'перше головне напруження
(МПа)
s2(i) = sn((N - 1) * 12 + 8) 'друге
s3(i) = sn((N - 1) * 12 + 9) 'третє
'Debug.Print i, s1(i), s2(i), s3(i)
Next i
Debug.Print N, FOS(s1(2), s1(1), s2(2), s2(1), s3(2),
s3(1))
Next N

End Sub
```

Об'єкти Matlab Automation Server Type Library

MATLAB - це комерційний пакет прикладних програм для числового аналізу, а також мова програмування, що використовується в цьому пакеті. MATLAB містить велику кількість функцій з різних областей математики (лінійна алгебра, статистика, обробка даних, диференціальні рівняння тощо). До цих функцій можна звернутись з VBA-програми за допомогою MATLAB Automation Server Type Library. В прикладі показано створення матриць в VBA-програмі, обчислення суми матриць в MATLAB і повернення результату в VBA-програму. Інші способи використання в VBA різних математичних процедур описані в [1, 3, 10, 17].

```
Dim MatLab As Object 'об'єкт MatLab
Dim b(1, 2) As Double 'масив (реальна частина)
Dim z(1, 2) As Double 'нульовий масив (уявна частина)

Public Sub main()
Set MatLab = CreateObject("Matlab.Application") 'створити
об'єкт MatLab
'заповнити масиви
```

```
For i = 0 To 1
    For j = 0 To 2
        b(i, j) = 1 'одиницями
        z(i, j) = 0 'нулями
    Next j
Next i
'виконати команди MatLab
MatLab.PutFullMatrix "b", "base", b, z 'передати матрицю
'b' в MatLab
MatLab.Execute "a = [1 2 3; 4 5 6]" 'створити матрицю 'a'
MatLab.Execute "b = a + b" 'додати матриці
'передати матрицю 'b' в VBA програму
MatLab.GetFullMatrix "b", "base", b, z
'вивести 'b'
Debug.Print b(0, 0), b(0, 1), b(0, 2)
Debug.Print b(1, 0), b(1, 1), b(1, 2)
MatLab.Quit 'вийти з MatLab
End Sub
```

Microsoft DAO - об'єктний доступ до даних

База даних - це організований набір даних, які доступні для машинної обробки. Реляційні бази даних складаються з таблиць, які містять поля (стовпчики) і рядки (записи). Для взаємодії з реляційними базами даних використовують предметно-орієнтовану мову структурованих запитів SQL. Microsoft Data Access Objects (DAO) - прикладний програмний інтерфейс для доступу до даних [29, 33, 45]. Приклад створює базу даних db1.mdb з інформацією про студентів і виконує SQL-запит до цієї бази даних. Перед виконанням виберіть в меню: Tools/References/Microsoft DAO 3.6 Object Library.

```
Dim РобочаОбласть As DAO.Workspace 'робоча область
Dim БазаДаних As DAO.Database 'база даних
Dim Таблиця As DAO.TableDef 'таблиця
Dim Поле As DAO.Field 'поле
```

```
Dim Запис As DAO.Recordset 'набір записів
Dim Sqlstr As String 'рядок для команд SQL

Public Sub main()
'Створюємо робочу область
Set РобочаОбласть = CreateWorkspace("", "admin", "",
dbUseJet)
'Створюємо базу даних "db1.mdb"
Set БазаДаних = РобочаОбласть.CreateDatabase("db1",
dbLangGeneral)
Set Таблиця = New DAO.TableDef 'створюємо таблицю
With Таблиця 'з таблицею
    .Fields.Append .CreateField("Прізвище", dbText)
'створити поле "Прізвище"
    .Fields.Append .CreateField("Оцінка", dbInteger)
'створити поле "Оцінка"
End With
Таблиця.name = "stud" 'ім'я таблиці
БазаДаних.TableDefs.Append Таблиця 'додати таблицю
БазаДаних.Close 'закрити базу даних
Set БазаДаних = РобочаОбласть.OpenDatabase("db1.mdb",
True) 'відкрити базу даних
'Створюємо записи таблиці "stud"
Set Запис = БазаДаних.OpenRecordset("stud", dbOpenDynaset)
Запис.AddNew 'додати запис
Запис.Fields("Прізвище").Value = "Іванов" 'ввести значення
поля "Прізвище"
Запис.Fields("Оцінка").Value = 4 'ввести значення поля
"Оцінка"
Запис.Update 'оновити
Запис.AddNew 'додати запис
Запис.Fields("Прізвище").Value = "Петров"
Запис.Fields("Оцінка").Value = 3
Запис.Update 'оновити
Запис.Close 'Закрити записи

'Команда SQL вибирає записи з таблиці stud, де Оцінка>3
```

```
Sqlstr = "SELECT * FROM stud WHERE (Оцінка>3)"
'Створюємо записи за запитом SQL
Set Запис = БазаДаних.OpenRecordset(Sqlstr, dbOpenDynaset)
'Виведення записів запиту
Запис.MoveFirst 'перейти на перший запис
Do While Not Запис.EOF 'поки не кінець записів
Debug.Print Запис.Fields("Прізвище").Value 'вивести
значення поля "Прізвище"
Debug.Print Запис.Fields("Оцінка").Value 'вивести значення
поля "Оцінка"
Запис.MoveNext 'перейти на наступний запис
Loop
'Закрити запис, базу даних і робочу область
Запис.Close
БазаДаних.Close
РобочаОбласть.Close
End Sub
```

Microsoft ADO - об'єктний доступ до даних

Microsoft ActiveX Data Objects (ADO) - прикладний програмний інтерфейс для доступу до даних, оснований на технології компонентів ActiveX, який є більш універсальним наступником технології DAO [19, 29, 32, 33, 42]. ADO дозволяє користувачам розробляти програми для доступу до різних баз даних (реляційних СУБД, текстових файлів тощо) без необхідності знання про те, як ці бази даних реалізовані. Основним об'єктом є `ADODB.Connection` - унікальний сеанс з джерелом даних. В системі бази даних клієнт-сервер він еквівалентний мережевому з’єднанню з сервером. За допомогою нього ви можете налагодити з’єднання з джерелом даних, вказати провайдера бази даних, установити і розірвати з’єднання. В прикладі відкривається джерело даних "db1.mdb", виконується запит мовою SQL і результати запиту зберігаються у файлі "db2.xml". Для запуску виберіть в меню Tools/References/Microsoft ActiveX Data Objects 2.7 Library.

```
Dim con As ADODB.Connection 'з'єднання
Dim rs As ADODB.Recordset 'набір записів

Public Sub main()
Set con = New ADODB.Connection 'створити з'єднання
con.Provider = "Microsoft.Jet.OLEDB.4.0" 'провайдер
con.ConnectionString = "db1.mdb" 'рядок з'єднання до
джерела даних
con.Open 'відкрити з'єднання
Set rs = New ADODB.Recordset 'створити набір записів
rs.Source = "SELECT * FROM stud WHERE (Оцінка>3)" 'рядок
запиту SQL
Set rs.ActiveConnection = con 'активне з'єднання
rs.Open 'відкрити набір записів
rs.MoveFirst 'перейти до першого запису
Do While Not rs.EOF 'поки не кінець записів
    Debug.Print rs.Fields(0).Value 'вивести значення
першого поля
    Debug.Print rs.Fields(1).Value 'вивести значення
другого поля
    rs.MoveNext 'перейти до наступного запису
Loop
rs.Save "db2.xml", adPersistXML 'зберегти як файл XML
rs.Close 'закрити набір записів
con.Close 'закрити з'єднання
End Sub
```

Об'єкти Msxml2.DOMDocument.5.0

Розширювана мова розмітки XML (Extensible Markup Language) - це стандарт побудови мов розмітки ієрархічно структурованих даних. Документ XML складається з елементів, наприклад `<el>Hello</el>`. Початок і кінець елемента позначається тегами `<el>` і `</el>`. Елемент може мати певний вміст (`Hello`). Елемент може також містити інші елементи. Прикладами мов, основаних на XML, є XHTML, SVG, MathML.

Об'єктна модель документа DOM (Document Object Model) - це специфікація прикладного програмного інтерфейсу для роботи з документами XML. DOM дозволяє створювати, читати і змінювати XML документи. DOM подає XML документи як деревовидну структуру, де кожен вузол є об'єктом, що відповідає частині документу. Msxml2.DOMDocument - це реалізація інтерфейсу DOM [19].

```
'Примітка: * - Microsoft розширення W3C DOM
Public Sub main()
Dim xmldoc As New MSXML2.DOMDocument50 'створити об'єкт
xml документ
'або так:
'Dim xmldoc As MSXML2.DOMDocument50
'Set xmldoc = CreateObject("Msxml2.DOMDocument.5.0")
Dim docNode, el1Node, el2Node, node As MSXML2.IXMLDOMNode
'об'єкти xml-вузли
Dim txtNode As MSXML2.IXMLDOMText 'об'єкт xml текстовий
вузол
Dim el As MSXML2.IXMLDOMElement 'об'єкт xml-елемент
Dim atr As MSXML2.IXMLDOMAttribute 'об'єкт xml-атрибут

'*завантажити документ з рядка
xmldoc.loadXML _
        "<?xml version='1.0'?>" + vbNewLine + _
        "<doc title='my'>" + vbNewLine + _
        "   <el1 atr1='1'>" + vbNewLine + _
        "      <el2 atr1='hello'>" + vbNewLine + _
        "          Hello world!" + vbNewLine + _
        "      </el2>" + vbNewLine + _
        "   </el1>" + vbNewLine + _
        "</doc>" + vbNewLine
xmldoc.Save "e:\\my.xml" '*зберегти документ

Dim b As Boolean
b = xmldoc.Load("e:\\my.xml") '*завантажити документ
If b Then 'якщо завантажено, тоді
```

```
    Set docNode = xmldoc.documentElement 'кореневий
елемент
    Set el1Node = docNode.FirstChild 'перший дочірній
вузол вузла docNode
    Set el2Node = el1Node.FirstChild 'перший дочірній
вузол вузла el1Node
    Set txtNode = el2Node.FirstChild 'перший дочірній
вузол вузла el2Node
    Debug.Print txtNode.NodeValue 'вивести текстове
значення вузла

    Set el = xmldoc.createElement("el3") 'створити вузол
елемента з іменем el3
    el.Text = "hello!!!" '*текстовий вміст вузла і
підвузлів
    Set atr = xmldoc.createAttribute("attr") 'створити
атрибут з іменем attr
    atr.Value = "10" 'значення атрибута
    el.setAttributeNode atr 'установити вузол атрибута atr
в елемент el
    docNode.appendChild el 'додати дочірній вузол як
останній

    Set node = docNode.LastChild.CloneNode(True)
'клонувати останній дочірній вузол з підвузлами
    docNode.InsertBefore node, el1Node 'вставити дочірній
вузол перед el1Node
    docNode.RemoveChild node 'видалити дочірній вузол node

    docNode.appendChild xmldoc.createTextNode("hello")
'додати дочірній текстовий вузол

    Dim s As Variant
    Set node = docNode.ChildNodes.Item(0) 'дочірній вузол
з індексом 0 (перший)
    s = node.nodeName 'ім'я вузла
    s = node.NodeType 'тип вузла
```

```
    s = node.NodeValue 'текстове значення вузла
    s = node.HasChildNodes 'чи вузол має дочірні вузли
    s = node.Text '*текстовий вміст вузла і підвузлів
    s = node.XML '*XML вузла і підвузлів
    s = node.DataType '*тип даних вузла
    s = node.parsed '*перевіряє чи вузол і підвузли
проаналізовані
    s = node.ParentNode.nodeName 'ім'я батьківського вузла
    s = node.NextSibling.nodeName 'ім'я наступного
спорідненого вузла
    s = node.ChildNodes.Item(0).nodeName 'ім'я дочірнього
вузла з індексом 0
    s = node.Attributes.Length 'кількість атрибутів вузла
    s = node.Attributes.Item(0).NodeValue 'текстове
значення атрибута з індексом 0
    s = node.Attributes.Item(0).specified '*чи заданий
явно, чи за замовчуванням
    s = docNode.ChildNodes.Item(0).OwnerDocument.nodeName
'ім'я кореня документа

    s = xmldoc.getElementsByTagName("el2").Length
'кількість елементів з тегом el2
    Set el = xmldoc.getElementsByTagName("el2").Item(0)
'перший елемент з тегом el2
    s = el.nodeName 'ім'я вузла
    s = el.tagName 'ім'я тега
    s = el.GetAttribute("atr1") 'значення атрибута atr1
    el.setAttributeNode xmldoc.createAttribute("Data")
'створити і установити атрибут Data
    el.setAttribute "Data", "today" 'задати значення
атрибуту Data
    s = el.getAttributeNode("Data").Value 'значення
атрибута Data
    Debug.Print s

    xmldoc.Save "e:\\my.xml" '*зберегти документ
End If
```

```
End Sub
```

```
<?xml version="1.0"?>
<doc title="my">
    <el1 atr1="1">
        <el2 atr1="hello" Data="today">
          Hello world!
        </el2>
    </el1>
    <el3 attr="10">hello!!!</el3>
hello
</doc>
```

Об'єкти Internet Explorer

Об'єктна модель браузера Internet Explorer дозволяє VBA-програмі звертатись до його компонентів, зокрема документа HTML [42]. Приклад створює об'єкт Internet Explorer, показує браузер та створює в ньому HTML-документ з таблицею (рис. 3.10).

```
Dim IE As Object 'об'єкт програма Internet Explorer

Public Sub main()
Set IE = CreateObject("InternetExplorer.Application")
'створити об'єкт
IE.Navigate "About:Blank" 'початкова сторінка
IE.Toolbar = False 'відключити панель інструментів
IE.StatusBar = False 'відключити рядок стану
Do
Loop While IE.Busy 'цикл "поки браузер зайнятий"
IE.Visible = True 'зробити видимим
'заголовок html документу
IE.Document.Write "<html><title>My table</title><body>"
IE.Document.Write "<p>Table 1</p>" 'абзац
IE.Document.Write "<table border=1>" 'таблиця
'заголовок таблиці
IE.Document.Write
"<tr><td><b>N</b></td><td><b>Name</b></td></tr>"
```

```
IE.Document.Write "<tr><td>1</td><td>Vasya</td></tr>"
'перший рядок
IE.Document.Write "<tr><td>2</td><td>Vova</td></tr>"
'другий рядок
IE.Document.Write "</table></body></html>" 'кінець таблиці
і документа
End Sub
```

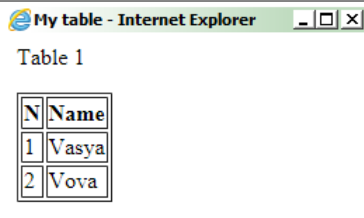


Рисунок 3.10 - HTML-документ в Internet Explorer

Об'єкти Microsoft Speech Object Library

Microsoft Speech Object Library містить об’єкти для роботи з синтезатором мовлення Windows (програмою для перетворення тексту в мовний сигнал). Приклад показує як можна легко озвучити вашу програму. Виберіть в меню Tools/References/Microsoft Speech Object Library.

```
Dim V As SpeechLib.SpVoice 'голос

Public Sub main()
Set V = New SpeechLib.SpVoice 'новий голос
Set V.Voice = V.GetVoices("Name=Microsoft Sam",
"Language=9").Item(0) 'параметри голосу
V.Speak "hello" 'сказати слово
End Sub
```

РОЗДІЛ 4. ЗАДАЧІ

1. Створити програму для обчислення значення функції *f(x)*, якщо *i*=1. Звірити результат з графіком (рис. 4.1).

$$f(x) = \frac{i + x}{ix + 2{,}5i} + \cos^{i+1}\left(x + \frac{\pi}{2}\right)$$

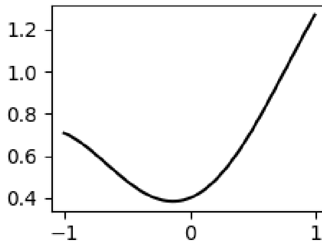


Рисунок 4.1 – Графік функції *f(x)*

2. Створити програму для обчислення значення функції *g(x)*, якщо *a*=-0,5; *b*=0,5.

$$g(x) = \begin{cases} 0, & x < a \\ f(x), & a \le x \le b \\ x, & x > b \end{cases}$$

3. Розв'язати попередню задачу з використанням підпрограми-функції, яка повертає значення *g(x)*.

4. Створити програму для виведення списку значень функції *g(x)*, якщо *x* змінюється від -1 до 1 з кроком 0,1. Використати оператори циклу `for`, `do` або `while`.

5. На основі попередньої програми зберегти значення аргументу в одновимірний масив X, значення функції в одновимірний масив Y. В двовимірний масив XY зберегти значення аргументу і функції. Знайти в цих масивах значення, які більші 0.

6. Знайти наближене значення інтеграла $\int_{-1}^{1} f(x)dx$ з кроком інтегрування 0,1. Використати оператори циклу.

7. Визначте тривалість обчислення значення функції *f(x)* і її інтеграла на вашому комп'ютері.

8. Знайти наближене мінімальне і максимальне значення функції *f(x)* на проміжку [-1, 1], якщо крок *x* рівний 0,1. Використати оператори циклу і умови.
9. Знайти наближено корінь рівняння *f(x)=1* на проміжку [-1,1], якщо крок *x* рівний 0,1. Використати оператори циклу і умови.
10. Створити словник, ключами якого є елементи (-1, 0, 1), а значеннями – відповідні значення функції *f(x)*. Вивести усі ключі і значення.
11. Створити довільний рядок. За допомогою операторів циклу і умови замінити задані символи (або слова) в рядку на інші. Розв'язати цю ж задачу за допомогою функції `Replace`.
12. Створити довільний рядок. За допомогою операторів циклу і умови замінити текст в дужках на три крапки (`...`). Розв'язати цю ж задачу за допомогою бібліотеки Microsoft VBScript Regular Expressions 5.5.
13. Дано множини A={1,2,3,4} і B={3,4,5,6}. Знайти об'єднання, перетин, різницю і симетричну різницю цих множин. Для збереження елементів множин можна використати Scripting.Dictionary.
14. Створити рекурсивну функцію *f(x,n)*, де *n* – додатне ціле число:

$$f(x,n) = \begin{cases} x^2, & n = 0 \\ f(x^2, \quad n-1), & n \neq 0. \end{cases}$$

15. Записати у текстовий файл значення аргументу *x* і функції *f(x)*, якщо *x* змінюється від -1 до 1 з кроком 0,1. Виконати пошук у створеному файлі значення *f(0)*.
16. Записати у бінарний файл довільного доступу значення аргументу *x* і функції *f(x)*. Виконати пошук у створеному файлі значення *f(0)*.
17. Від дати 2019-01-02 02:17:01 відняли 1e6 секунд. Яка це дата? Який це день тижня?
18. Створити клас, який містить змінні (властивості) *xmin*, *xmax*, *dx* і методи *f(x)*, *X()*, *F()*, які, відповідно, повертають значення функції, список значень аргументу *x*, список значень функції *f(x)*. Створити об'єкти класу.

19. Створити клас, який містить методи, що повертають значення функції $f(x_0)$, похідної $f'(x_0) = df(x_0)/dx \approx (f(x_0 + \Delta x) - f(x_0))/\Delta x$ і інтеграла $F(x_0) = \int_{x_{\min}}^{x_0} f(x)dx \approx \sum_{i=1}^{n} f(x_i)\Delta x$ в заданій точці x_0, де $n = (x_0 - x_{min})/\Delta x$.

20. Створити клас, який описує поняття вектора $\{a_1, a_2, a_3\}$ і містить функції для обчислення його довжини $\sqrt{a_1^2 + a_2^2 + a_3^2}$, додавання двох векторів $c_i = a_i + b_i$, скалярного множення $c = \sum_{i=1}^{3} a_i b_j$ і векторного множення:

$$c = a \times b = \begin{Bmatrix} a_2 \cdot b_3 - a_3 \cdot b_2 \\ a_3 \cdot b_1 - a_1 \cdot b_3 \\ a_1 \cdot b_2 - a_2 \cdot b_1 \end{Bmatrix}.$$

21. Розробити бібліотеку DLL мовою C з функцією, що повертає список значень функції *f(x)*. Викликати цю функцію з VBA.

22. Розробити програму з графічним інтерфейсом для виведення значення функції *f(x)*. Передбачити обробку виняткових ситуацій, які виникають під час введення недопустимих значень аргументу.

23. Розробити програму з графічним інтерфейсом для виведення списку значень функції *f(x)*. Використати такі елементи керування як `CommandButton`, `Label`, `TextBox`, `CheckBox`, `OptionButton`, `ListBox`, `ScrollBar`.

24. З програми VBA записати у два стовчики листа Excel значення аргументу *x* і функції *f(x)*, якщо *x* змінюється від -1 до 1 з кроком 0,1. Виконати пошук у створеному листі значення *f(0)*.

25. В VBA-програмі згенерувати випадкову вибірку на листі Excel. Обчислити емпіричні середнє значення та середньоквадратичне відхилення шляхом виклику функцій Excel.

26. З програми VBA побудувати графік функції *f(x)* за допомогою `Chart`.

27. За допомогою `Chart` знайти регресію виду $f(x) = ax^2 + bx + c$ за даними *x*, *y* (табл. 4.1). Знайти коефіцієнт детермінації R^2.

Таблиця 4.1 - Емпіричні дані - залежність *y* від *x*

x	0	1	2	3
y	0	1	4	10

28. Створити програму, яка додає в документ Word значення перших двох комірок Excel і виділяє перше слово жирним шрифтом.

29. З VBA-програми зберегти у текстовий файл список елементів поточного каталогу і вкладених каталогів. Визначити розмір їх файлів в байтах. Створити новий каталог і скопіювати в нього цей текстовий файл.

30. За допомогою SOLIDWORKS API змінити розміри довільної параметричної моделі SOLIDWORKS, перебудови її і обчислити об'єм тіла.

31. За допомогою SOLIDWORKS API змінити рівняння довільної параметричної моделі SOLIDWORKS з рівняннями, перебудови її і обчислити моменти інерції.

32. За допомогою SOLIDWORKS Simulation API створити програму для пошуку значення параметра моделі (в заданій множині значень), яке спричиняє в ній мінімальні напруження.

33. Розв'язати задачі про пошук коренів рівняння *f(x)=1*, мінімуму функції *f(x)* і її інтеграла шляхом доступу до об'єктів MATLAB з програми VBA.

34. Розв'язати систему лінійних рівнянь за допомогою об'єктів MATLAB:

$$\begin{cases} a + b + c = 4, \\ 2a - b + 2c = 2, \\ a + 2b - c = 4. \end{cases}$$

35. Дано рядок, який містить підрядки <ключ>=<значення>. Зберегти усі ключі і значення у базу даних. Прочитати з бази даних значення за заданим ключем. Використати Microsoft DAO або ADO.

36. Створити програму, яка записує значення функції *f(x)* у XML-документ в такому вигляді: `<point><x>0.1</x><y>-1.25</y></point>`. Використати Msxml2.DOMDocument.
37. Створити програму, яка шукає в XML-документі з попередньої задачі значення *y>0*. Використати Msxml2.DOMDocument.

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1 ALGLIB Free Edition. Available at: www.alglib.net/download.php
2 Autodesk Inventor API. Первые шаги. Режим доступа: https://ru.wikibooks.org/wiki/Autodesk_Inventor_API._ Первые_шаги
3 Billo E. Joseph. Excel for Scientists and Engineers: Numerical Methods. - Wiley, 2007. - 477p.
4 Excel 2003 VBA Programmer's Reference / Paul Kimmel, Stephen Bullen, John Green, Rob Bovey, Robert Rosenberg. - Indianapolis : Wiley, 2004. - 1177p.
5 Getz K. VBA Developer's Handbook, Second Edition / Ken Getz, Mike Gilbert. - SYBEX, 2000. - 1104p.
6 Hansen S. Mastering Excel 2003 Programming with VBA. - SYBEX, 2004. - 606p.
7 Himpe Vincent. Visual Basic for Electronics Engineering Applications. - 2002. - 569 p.
8 Malpass L. SolidWorks 2008 API Programming & Automation. - AngleSix, 2013. - 267 p.
9 Malpass L. SolidWorks 2009 API Advanced Product Development. - AngleSix, 2013. - 246 p.
10 NAG NUMERICAL LIBRARIES AND MICROSOFT VISUAL BASIC. Available at: https://www.nag.com/languages-and-environments
11 Professional Excel Development: The Definitive Guide to Developing Applications Using Microsoft® Excel and VBA® / Stephen Bullen, Rob Bovey, John Green. - Addison Wesley Professional, 2005. - 936.
12 Spens Mike. Automating SolidWorks 2011 Using Macros - SDC, 2010. - 398 p.
13 The Future of VBA Programming. Available at: http://www.softwaretogo.de/blog/programming-languages-101-vba
14 Автоматизоване проектування різальних інструментів: Навчальний посібник / В.Б. Копей, О.Р. Онисько, Л.О. Борущак, Л.Я. Роп'як. - Івано-Франківськ : ІФНТУНГ, 2012. - 208с.

15 Берндт Г., Каинка Б. Измерение, управление и регулирование с помощью макросов VBA в Word и Excel. - К.: "МК-Пресс", 2008. - 256 с.
16 Биллиг В.А. VBA в Office 2000. Офисное программирование. — М.: Издательско-торговый дом «Русская Редакция» , 1999. — 480 с.: ил.
17 Гайдишев И.П. Решение научных и инженерных задач средствами Excel, VBA и C/C++. - СПб.: БХВ-Петербург, 2004. - 512 с.
18 Гарбер Г. З. Основы программирования на Visual Basic и VBA в Exce1 2007. —М.: СОЛОН-ПРЕСС, 2008. — 192 с.
19 Гарнаев А. Ю. Самоучитель VBA. Второе издание. СПб: BHV, 2004. - 542 с.
20 Гетц К, Гилберт М. Программирование на Visual Basic 6 и VBA. Руководство разработчика: Пер. с англ. - К.: BHV, 2001. - 912 с.
21 Гетц К, Джилберт М. Программирование в Microsoft Office. Для пользователя: пер. с англ. — К.: Издательская группа BHV, 2000. — 384 с.
22 Глинський, Я. М. Бейсик. QBasic, Visual Basic і VBA: навч. посіб. / Я. М. Глинський, В. Є. Анохін, В. А. Ряжська. - 4-те доп. вид. - Львів : Деол, 2004. - 158 с.
23 Демидова Л.А., Пылькин А.Н. Программирование в среде Visual Basic for Applications: Практикум. — М.: Горячая линия - Телеком, 2004. — 175 с.
24 Климачева Т. Н. Трехмерная компьютерная графика и автоматизация проектирования в AutoCAD 2007. – М.: ДМК, 2007. – 464 с., ил.
25 Кондрашов, Ю. Н. Программирование на Visual Basic для Windows 95 / Ю. Н. Кондрашов. - М. : Радио и связь, 1997. - 256 с.
26 Копей, В.Б. Методичні рекомендації з дисципліни «Прикладне програмування» / В.Б. Копей. - Івано-Франківськ : ВМУРоЛ «Україна» Івано-Франківська філія: "ІНІН", 2006. - 46с.
27 Копей, В.Б. Система тривимірного параметричного проектування SolidWorks.: Методичні вказівки для вивчення

дисциплін “Основи наукових досліджень” і “Принципи інженерної творчості” / В.Б. Копей. - Івано-Франківськ : Факел, 2005. - 45с.
28 Кудрявцев Н.Г. Программирование на VBA MS Excel: учебное пособие / Н. Г. Кудрявцев, Д. В. Кудин, М. Ю. Беликова. - Горно-Алтайск: РИО ГАГУ, 2015. - 116с.
29 Кузьменко В. Г. Программирование на VBA 2002. — М.: ООО «Бином-Пресс», 2003 —880 с.: ил.
30 Культин Н. Б. Visual Basic. Освой на примерах. — СПб.: БХВ-Петербург, 2004. - 288 с: ил.
31 Малышев С.А.Самоучитель VBA. Как это делается в Word, Exce1, Access. — СПб: Наука и Техника, 2001 — 496 с.
32 Михеев Р. VBA и программирование в Microsoft Office для пользователей. - СПб: BHV, 2006. - 361с.
33 Программирование в пакетах MS Office: учеб. пособие / С.В. Назаров, Л.П. Мельников, Л.П. Смольников и др.; под ред. С.В. Назарова. — М.: Финансы и статистика, 2007. —656 с.
34 Роман С. Использование макросов в Exce1. 2-е изд. — СПб.: Питер, 2004. —507 с.: ил.
35 Симонович С. Занимательное программирование Visual Basic. / С. Симонович , Г. Евсеев. - М: Аст-Пресс, 2001. - 320 с.
36 Слепцова Л.. Программирование на VBA. Самоучитель . Издательский дом « Диалектика-Вильямс », 2004. - 384 с .
37 Справочник по VBA для Office. Режим доступа: https://docs.microsoft.com/ru-ru/office/vba/api/overview
38 Стивенс Р. Visual Basic. Готовые алгоритмы: пер. с англ. / Род Стивенс. - М.: ДМК Пресс, 2000. - 384с.
39 Титов В. Л. Программирование на VBA Учебно методичесоке пособие - Могилев: Могилевский государственный университет продовольствия, 2008. - 124 с.
40 Уокенбах, Джон. Microsoft Office Excel 2007. Библия пользователя.: Пер. с англ. — М.: Вильямс, 2008. — 816 с.: ил.
41 Уокенбах, Джон. Профессиональное программирование на VBA в Excel 2003. : Пер. с англ. – М. : Издательский дом “Вильямс”, 2005. – 800 с. : ил.

альворсон М. Microsoft Visual Basic 6.0 для профессионалов. Шаг за шагом. Перевод с англ. - М.: Издательство ЭКОМ, 1999. - 720 с.
43 Харрис Мэтью. Освой самостоятельно программирование для Microsoft Excel 2000 за 21 день. Пер. с англ. : Учебное пособие. — М. : Издательский дом "Вильямс", 2000. — 880 с.
44 Хорев В.Д. Самоучитель программирования на VBA в Microsoft Office. — К.: Юниор, 2001. — 320 с., ил.
45 Штайнер Г. VBA 6.3. — М.: Лаборатория Базовых знаний, 2002. — 784 с.: ил. — (Справочник).
46 Эйткен, Питер. Интенсивный курс программирования в Excel за выходные.: Пер. с англ. — М.: Издательский дом "Вильямс", 2004. — 432 с.